



BEA WebLogic Platform™

Deploying WebLogic Platform Applications

Copyright

Copyright © 2004-2006 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, BEA JRockit, BEA Liquid Data for WebLogic, BEA WebLogic Server, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA MessageQ, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic JRockit, BEA WebLogic Log Central, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic WorkGroup Edition, BEA WebLogic Workshop, and Liquid Computing are trademarks of BEA Systems, Inc. BEA Mission Critical Support is a service mark of BEA Systems, Inc. All other company and product names may be the subject of intellectual property rights reserved by third parties.

All other trademarks are the property of their respective companies.

Contents

1. Overview of WebLogic Platform Deployment

About WebLogic Platform Applications	1-1
Promoting a WebLogic Platform Application from Development to Production	1-2
Comparing Development and Production Environments	1-3
Understanding the Application Promotion Process	1-4
How WebLogic Platform Supports Application Promotion	1-4
Planning the Promotion	1-5
Summary of Tasks for Promoting a WebLogic Platform Application to Production .	1-7
Automating the Promotion Process	1-8
Where to Find Additional Information	1-11

2. Understanding the Target Environment

Typical Deployment Environment	2-2
Considerations for Configuring WebLogic Platform Domains	2-4
WebLogic Platform Domain Examples	2-6
Development Domain Example	2-7
Components Shown in Example	2-7
Considerations for this Configuration	2-8
Single-Cluster Platform Domain Example	2-9
Components Shown in Example	2-9
Considerations for this Configuration	2-11
Multi-Domain Example	2-14

Components Shown in Example.	2-15
Considerations for this Configuration.	2-16
Multi-Cluster Platform Domain Example	2-17
Components Shown in Example.	2-18
Configuration Considerations.	2-19

3. Creating and Configuring the WebLogic Domain

Tools for Configuring the Target Domain	3-2
Considerations for Configuring and Targeting Resources	3-3
Guidelines for Specifying the Names and Addresses of the Members in a Cluster	3-12
Autoconfiguration Using the Configuration Wizard and WLST Offline	3-12
Adding Application Resources Required by the WebLogic Workshop Runtime.	3-13
Configuring Servers to Start in Production Mode.	3-14
Setting the SDK.	3-14
Setting Up the Managed Servers on Remote Machines	3-14
Setting Up the Managed Server Directories	3-15
Configuring the Managed Server Start Attributes	3-16
Configuring Node Manager	3-21
Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline.	3-21
Example: How to Configure a Multi-Domain Environment Using WLST Offline	3-28
Example: How to Configure a Multi-Domain Environment Using the Configuration Wizard in Silent Mode	3-29
Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline .	3-30

4. Configuring the Production Database

Creating and Preparing a Production Database.	4-1
Creating Conversational State Database Tables	4-2
Promoting Database Information to the Production Database	4-3
Promoting WebLogic Portal Datasync Information.	4-3

Promoting LDAP and Portal Database Data	4-5
Promoting WebLogic Integration Application Database Information	4-5
Example: How to Load the Domain Database Using WLST Offline	4-6
Example: How to Load the Application Database Using Ant	4-7
Example: How to Create the Conversational State Database Tables Using Ant	4-9

5. Configuring Security

Ways to Secure the Target Environment	5-2
Considerations for Configuring Security	5-6
Promoting Embedded LDAP Security Data to the Target Database	5-7
Promoting Security Data to the Target Environment	5-7
Promoting WebLogic Portal and WebLogic Integration Data	5-8
Maintaining Security Policy Files Under Version Control	5-8

6. Using Load Balancers and Web Proxy Servers

Load Balancing with an External Hardware Load Balancer	6-1
Load Balancing with a Web Proxy Server	6-2
Considerations When Configuring Load Balancers and Web Proxy Servers	6-3

7. Preparing the Application for Deployment

Summary of Application Preparation Tasks	7-2
Summary of Changes to Deployment Descriptors and Configuration Files	7-3
Managing Application Files for Promotion to Production	7-7
Access to Application Files	7-7
Using a Version Control System	7-8
Files that Should Not Be Under Version Control	7-9
Enabling High Availability	7-10
Enabling Inter-Application Communication	7-11
Example Web Service Control URL	7-12

Techniques for Updating URLs in Web Service Controls	7-12
Enabling Session Replication.	7-14
Specifying a Concurrency Strategy for Stateful Business Process Entity Beans. . .	7-14
Reconfiguring Application Views and Adapters	7-15
Reconfiguring Application Views	7-15
Specifying Event Generator Targets	7-16
Preparing Application Security	7-16
Limiting Access to Authenticated Users	7-16
Restricting Application Access to SSL Traffic Only.	7-17
Packaging the Application	7-18
Updating the Server Path Attribute	7-18
Generating the EAR File	7-19

8. Deploying the Application

About Deployment Units	8-1
Overview of the Deployment Tools	8-2
Deployment Considerations	8-3
Archive Type	8-3
Application Targets	8-3
Security Roles	8-4
Staging Modes	8-5
Load Order	8-6
Deployment Descriptors	8-6
Steps to Deploy the Application	8-7
Step 1: Start the Servers	8-7
Step 2: Upload the Application to the Administration Server (Optional)	8-7
Step 3: Deploy the Application	8-8

Deploying a WebLogic Integration, WebLogic Server, or WebLogic Workshop Application.	8-8
Deploying a WebLogic Platform or WebLogic Portal Application	8-8
Step 4: Deploy Event Generators	8-9
Starting the Servers	8-9
Before You Start the Servers	8-9
Starting the Administration Server.	8-10
Starting the Managed Servers.	8-11
Example: How to Deploy a WebLogic Integration Application Using weblogic.Deployer and Ant	8-11
Example: How to Deploy WebLogic Platform, WebLogic Portal, and WebLogic Integration Applications Using weblogic.Deployer and Ant	8-13

A. Deployment Targeting Reference

Characteristics of a Production Deployment Domain	A-1
Distribution of Production Domain Resources	A-2
Deployment Targeting Reference - Single Cluster WebLogic Platform Domain.	A-4
Default Domain Resource Reference - By Product Component	A-14

B. Deployment Checklists

Checklist for Planning the Promotion	B-2
Checklist for Installation and Network Configuration Requirements	B-5
Checklist for Creating and Configuring a WebLogic Domain	B-7
Checklist for Configuring a Production Database.	B-8
Checklist for Configuring Security	B-9
Checklist for Using Load Balancers and Web Proxy Servers	B-13
Checklist for Preparing Application Files	B-14
Checklist for Deploying the Application.	B-16

Overview of WebLogic Platform Deployment

This document explains how to deploy an application into a production environment that is based on WebLogic Platform. It describes the steps necessary to prepare the production environment and the application for deployment.

This overview includes the following topics:

- [About WebLogic Platform Applications](#)
- [Promoting a WebLogic Platform Application from Development to Production](#)
- [Where to Find Additional Information](#)

About WebLogic Platform Applications

Throughout this document, the term *WebLogic Platform application* refers to an application that is developed in WebLogic Workshop and that combines all components of WebLogic Platform. For example, an application that combines a WebLogic Integration business process with the WebLogic Portal rules engine is a WebLogic Platform application. Similarly, an application that combines a WebLogic Portal application with the Worklist component and the data transformation capability from WebLogic Integration is a WebLogic Platform application.

Structurally, a WebLogic Platform application is a Web application that complies with the J2EE specification and can be deployed to WebLogic Server. A WebLogic Platform application may encompass multiple Workshop projects and includes:

- The WebLogic Workshop application deployment descriptors, `web.xml` (J2EE), and `weblogic.xml` (WebLogic Server)

- Optionally one or more standalone modules—for example, Web services, EJBs, or resource adapters—and startup and shutdown classes

Deploying a WebLogic Platform application into a production environment requires:

1. A domain created with the Basic WebLogic Platform Domain template.

A domain is the basic administration unit for WebLogic Server. For more detailed information about WebLogic Server domains, see [“Overview of WebLogic Server Domains”](#) in *Configuring and Managing WebLogic Server*.

The template you use to create a domain defines the structure of each WebLogic Server instance that is configured in that domain; specifically, it determines the kinds of applications that can be deployed and run on those servers. Servers configured in a domain that is based on the Basic WebLogic Platform Domain template can run any kind of application supported in WebLogic Platform, including a WebLogic Platform application, assuming that the full WebLogic Platform software has been installed on each machine in the domain.

2. Selective targeting of modules in the WebLogic Platform application.

Due to specific targeting requirements for WebLogic Integration and WebLogic Portal (described in detail later in this document), when you deploy a WebLogic Platform application into a clustered environment, you typically need to target some modules only on the Administration Server, and other modules only on the cluster.

Depending on the characteristics of the environment in which a WebLogic Platform application is deployed, and of the application itself, there are other considerations as well; for example, considerations for configuring databases, security, and load balancers. These considerations are summarized in [Chapter 2, “Understanding the Target Environment.”](#)

Promoting a WebLogic Platform Application from Development to Production

This section discusses the following topics about promoting a WebLogic Platform application to a production environment:

- [Comparing Development and Production Environments](#)
- [Understanding the Application Promotion Process](#)
- [How WebLogic Platform Supports Application Promotion](#)
- [Planning the Promotion](#)

- [Summary of Tasks for Promoting a WebLogic Platform Application to Production](#)
- [Automating the Promotion Process](#)

Comparing Development and Production Environments

In a WebLogic *development environment*, developers create applications using WebLogic Workshop on a single instance of WebLogic Server. WebLogic Workshop facilitates iterative development by automatically generating server resources. The security configuration in a development environment is generally unrestricted.

In contrast, the *production environment* is typically a tightly controlled environment that may include multiple servers in one or more clusters. You need to configure resources required by an application separately and manually, although you may employ a rich set of configuration tools and scripts to construct a highly automated and controlled process to configure the resources. In addition, security and high-availability are critical in a production environment.

The following table compares typical differences between a development environment and a production environment in key areas.

Table 1-1 Comparison of the Development and Production Environments in Key Areas

Key Area	Development	Production
Domain environment	Single domain, single server, and single machine (no cluster)	Multiple domains; clustered servers on multiple machines; Node Manager used to manage server availability
Security	Embedded LDAP using demonstration digital certificates that enable anonymous access	Use of security information imported from a QA/Test environment; user and group information on external LDAP server. Managed Servers configured for SSL.
Database Stores	Developer-level database (e.g., PointBase) located on development machine	Enterprise-level database
Web proxy server/Load Balancer	Not required in an unclustered environment	Used to load balance, access information behind a firewall, enable secure access to multiple domains, and provide inter-application communication
Deployment Process	Application auto-deploy enabled	Automated deployment of applications using WebLogic Server tools and scripts

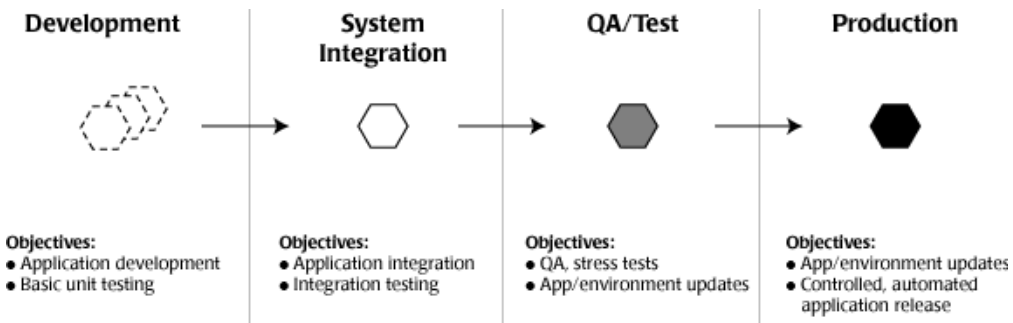
Because of the significant differences between the two environments, and the likely test environment in between, moving your application from development to production is typically a multi-stage process, as described in the next section.

Understanding the Application Promotion Process

Promoting an application from a development environment to an enterprise production environment can be a complex process requiring a fair amount of planning, preparation, coordination, and testing. It is a multi-step process: the application typically is deployed into one or more intermediate environments where it is integrated with other applications, undergoes rigorous testing, and runs in configurations that are scaled up or in which application resources are increasingly distributed. This document refers to this multi-step process toward production as the *promotion process*, or *application promotion*.

For example, the promotion process may consist of promoting an application from development to production in two or more stages. The following figure depicts a promotion process that moves the application through four stages.

Figure 1-1 Example Set of Stages in the Application Promotion Process



It is critical for teams to adopt a process for promoting an application production to ensure a successful production system deployment, as described later in this section.

How WebLogic Platform Supports Application Promotion

WebLogic Platform supports the application promotion process by providing a rich set of tools for creating and configuring the target application environment, building and deploying applications, and propagating application and security data. Most of these tools provide both a graphical user interface and scripting interface for ease-of-use and automation. The script-based tools give you the ability to automate application promotion.

While these tools facilitate the promotion process, they cannot take the place of a comprehensive plan. The section that follows discusses application promotion planning in more detail.

Planning the Promotion

Planning the application promotion process from development to production is critical. Planning how you will promote an application from development to production helps to ensure a successful production deployment. This planning spans two broad areas:

- Defining the intermediate stages in which the application is to be promoted from development to production, and what changes are made at each stage.
- Identifying how each the environment will be configured at each stage, and the resources required to support the use and testing of the application in each stage.

When members of development, testing, and production teams collaborate to create a promotion plan, the plan has the buy-in, understanding, and sponsorship of all the people needed to create a reliable and reproducible production deployment.

A promotion plan includes detailed descriptions of the setup tasks and requirements for each environment in which the application is developed and deployed. Those involved in the development and deployment of the application collaborate to create a promotion plan. Detailed checklists are provided in this document, in [“Deployment Checklists” on page B-1](#), to facilitate the creation of an application promotion plan.

In short, the promotion plan is meant to address the following:

- Hardware and software requirements of each target environment, including:
 - The complete set of hardware required at each stage, including the machines hosting the database, load balancers, firewalls, etc., including IP addresses, DNS names, and ports, as appropriate
 - Network requirements
 - Software requirements, including WebLogic Platform, operating systems, third-party software, software version control tools, etc.
- Domain configuration requirements, including:
 - Machine addresses, DNS names, and listening ports for hosting the domain, specifically the Administration Server, Managed Servers, and cluster
 - Configuration information for Relational Database Management System (RDBMS) used

- Node Manager requirements
- Shared file system requirements, such as Storage Area Network
- SDK requirements
- Application requirements for system-level resources, such as: JMS for reliable messaging; persistent storage for conversational Web services; logging
- Cluster requirements, including replication groups, load balancing for EJBs
- Security requirements, including:
 - Required WebLogic security providers
 - LDAP server requirements
 - List of resources that need to be secured, including security policies required to protect them
 - SSL requirements, including certificates and how they are stored, identity of Certificate Authority, etc.
 - Action on Security advisories that are required
- Application file requirements, such as:
 - Version control requirements; for example, identifying files to maintain in a version control tool such as Perforce® or CVS.
 - Source code changes required for the target environment, such as for enabling high availability or specific security requirements
 - Deployment descriptor changes
 - Configuration changes required for WebLogic Integration application views
- Application packaging and deployment requirements, such as:
 - Scripts for automating various tasks
 - Library files, JAR files, etc. that need to be included
 - Deployment format requirements, such as compressed vs. exploded EAR format, stage vs. nostage
 - Deployment order requirements
- Configuration tools, application build tools, and deployment tools to be used; for example, Configuration Wizard, WLST Offline, `wlwBuild`, `weblogic.deploy`, source control management system

- A test plan, which specifies the set of unit, stress, or other QA test requirements for each promotion stage

Just as importantly, a promotion plan also identifies the team members who will be involved in each stage of promotion, and their specific roles, tasks, and responsibilities at each stage.

Summary of Tasks for Promoting a WebLogic Platform Application to Production

The following table summarizes the tasks that need to be performed to promote an application from a development environment into a production environment.

Table 1-2 Summary of Tasks for Promoting a WebLogic Platform Application to Production

Step	Description
Installation and Network Requirements	Install WebLogic Platform in a secure manner on all machines targeted for use in the domain, and procure a product cluster production license for each required installation. Set up the network in a secure manner. For more information, see “Checklist for Installation and Network Configuration Requirements” on page B-5.
“Creating and Configuring the WebLogic Domain” on page 3-1	Create the WebLogic domain and set up the remote Managed Servers.
“Configuring the Production Database” on page 4-1	Set up the RDBMS for the production environment and populate it with data.
“Configuring Security” on page 5-1	Secure the target environment and promote embedded LDAP security data to the target database. Import security information from the prior environment, as appropriate; for example, user and group information, and policies set on applications and resources.
“Using Load Balancers and Web Proxy Servers” on page 6-1	Use a load balancer or Web proxy server if the target environment includes a cluster. A load balancer/Web proxy server distributes client connection requests, provides load balancing and failover across the cluster, and provides security by concealing the local area network addresses from external users.

Table 1-2 Summary of Tasks for Promoting a WebLogic Platform Application to Production (Continued)

Step	Description
“Preparing the Application for Deployment” on page 7-1	Modify application files so that the application can invoke other applications that may have different addresses, and to transition the application from running in non-clustered to a clustered environment. Comply with security requirements of the new environment, such as SSL.
“Deploying the Application” on page 8-1	Start the servers and deploy the application.

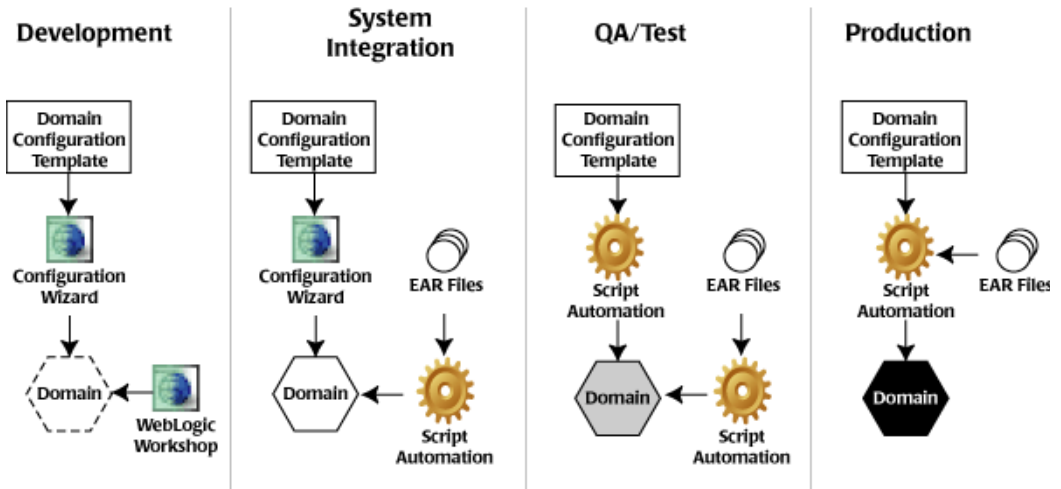
Automating the Promotion Process

Before deploying an application to a production environment, or to one of the intermediate stages, you should configure the environment in a careful and controlled manner. Automating the steps in the promotion process via the use of tools and scripts provide the following advantages:

- **Reproducibility**—Enables you to reproduce an environment.
- **Reliability**—Ensures that you are executing a set of steps that have been tested, and eliminates errors introduced by manual operation.
- **Ease-of-use**—Enables you to deploy applications quickly and easily by executing a set of scripts.

The following figure illustrates how automation might be introduced at each stage of the promotion process. As shown, the level of automation increases at each stage. Scripts that are tested and proven in one stage can be applied in subsequent stages. In this example, the production stage is fully automated.

Figure 1-2 Example of Automating the Promotion Process



It is recommended that the directory structure be identical across all stages to simplify the creation and re-use of configuration templates and shell scripts.

The following table defines the WebLogic tools available to automate the process. These tools can be used in conjunction with Ant or other shell scripting tools.

Table 1-3 WebLogic Tools Used for Automating the Promotion Process

The following tool(s)...	Enables you to...
WebLogic Scripting Tool (WLST)	<p>Perform administration tasks and initiate WebLogic configuration changes interactively or by using an executable script.</p> <p>There are two versions of WLST available from BEA’s dev2dev site:</p> <ul style="list-style-type: none"> <p>WLST Offline—Enables you to create a new domain or update an existing domain without connecting to a running WebLogic Server—supporting the same functionality as the Configuration Wizard. Instructions for setting up and using WLST offline, as well as sample scripts for configuring domains, are available from dev2dev code sample site at the following URL:</p> <p>https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97</p> <p>WLST Online—Enables you to perform administrative tasks and initiate WebLogic Server configuration changes while connected to a running server. Instructions for setting up and using WLST online, as well as sample scripts for configuring WebLogic Server, are available from dev2dev code sample site at the following URL:</p> <p>https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=13</p> <p>Examples are provided in this document to illustrate automation of key Platform deployment steps using WLST Offline.</p>
Configuration Wizard Silent-mode Configuration	<p>Execute the Configuration Wizard as a noninteractive process. The Configuration Wizard reads configuration settings from a script that you manually create prior to execution.</p> <p>Note: This tool and the WLST Offline tool support the same functionality.</p> <p>For more information about silent-mode configuration, see Creating WebLogic Configurations Using the Configuration Wizard.</p>
WebLogic Server Command-Line Interface	<p>Manage a WebLogic Server domain by issuing commands manually on the WebLogic Server command line. This method enables you to manage a domain without the Administration Console, when use of the console is neither practical nor desirable. For more information, see “weblogic.Admin Command-Line Reference” in the <i>WebLogic Server Command Reference</i>.</p>

Table 1-3 WebLogic Tools Used for Automating the Promotion Process (Continued)

The following tool(s)...	Enables you to...
Java Utilities	Perform common tasks, such as deploying applications and testing DBMS configurations. For more information, see “ Using the WebLogic Server Java Utilities ” in the <i>WebLogic Server Command Reference</i> .
Ant Tasks	Create powerful build scripts. For more information, see: <ul style="list-style-type: none"> • “Using Ant Tasks to Configure a WebLogic Server Domain” in the <i>WebLogic Server Command Reference</i>. • The Apache Ant Project Web site at: http://ant.apache.org
Deployment Tools	Deploy applications and standalone modules to WebLogic Server. For more information, see “ Deployment Tools Reference ” in <i>Deploying WebLogic Server Applications</i> .
WebLogic Portal Propagation Utility and Datasync Web Application	Promote LDAP and portal database data. For more information, see “ Using the Propagation Utility ” and “ Using the Datasync Web Application ” in the <i>WebLogic Portal Production Operations User Guide</i> .
WebLogic Integration Bulk Loader	Import, export, and delete trading partner data, including trading partner profiles, certificates from keystores, service definitions, and service profiles. For more information, see “ Using the Trading Partner Bulk Loader ” in <i>Managing WebLogic Integration Solutions</i> .

The examples provided throughout this document illustrate how you can automate the promotion process using a subset of the tools listed above.

Where to Find Additional Information

This document supplements the information provided in the following guides, which describe the deployment considerations for applications that are built using a single WebLogic Platform component, such as WebLogic Portal or WebLogic Integration. You may wish to reference them for more information at points indicated throughout this document.

- *Deploying WebLogic Server Applications*—describes how to deploy WebLogic Server applications in a production environment.
- *Production Operations User Guide*—describes how to manage the portal life cycle, from portal development to staging and testing to live production environments.

- *Deploying WebLogic Integration Solutions*—describes how to deploy WebLogic Integration solutions in a production environment.
- “*Deploying an Application to a Production Server*” in *WebLogic Workshop Online Help*—describes the basic concepts of deploying WebLogic Workshop applications in a production mode.

For a set of sample WLST scripts demonstrating how to configure WebLogic domain resources using WLST, see <https://wlst.projects.dev2dev.bea.com>.

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

For an illustration of how to promote and extend the WebLogic Platform Tour from a development to a clustered production environment, review *Clusterizing End2End on WebLogic Platform 8.1*, available on the dev2dev Web site at the following URL:

http://dev2dev.bea.com/products/wlplatform81/articles/clusterizing_E2E.jsp

Note: The Tour is included with the WebLogic Platform installation. For more information about the Tour, see the *WebLogic Platform Tour Guide*.

Understanding the Target Environment

The application promotion process is influenced not only by the application's design, but also by the hardware and software characteristics of the target environment in which the application will run. These characteristics include the machines, the operating systems on those machines, the network, databases, clusters, load balancers, and more, that host or significantly interact with the application.

This chapter describes four different environments into which WebLogic Platform applications are deployed, gives considerations regarding how the characteristics of each environment influence how the domains for each environment need to be configured, and identifies how application files may need to be modified prior to deployment into each environment. Specific considerations regarding security, high availability, and database usage are also provided.

The following topics are included:

- [Typical Deployment Environment](#)
- [Considerations for Configuring WebLogic Platform Domains](#)
- [WebLogic Platform Domain Examples](#)

The information provided in this chapter is at a conceptual and summary level only. For a detailed reference of the targeting and deployment of system-level resources, services, and applications in a WebLogic Platform domain, see [Appendix A, “Deployment Targeting Reference.”](#)

Typical Deployment Environment

A basic deployment environment based on a WebLogic domain typically includes the components listed in the following table. This table also shows the graphical symbols that represent these components in the illustrations throughout this chapter.

Table 2-1 Components of a Basic Application Deployment Environment





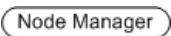
Component	Description
Domain 	The basic organizational unit in which the deployed resources are administered. It consists of one or more WebLogic Server instances, and logically related resources and services that are managed, collectively, as one unit.
Machine 	Hosts the WebLogic software, which is installed on the machine and on which WebLogic Server instances run, applications, database and RDBMS, and other software used in the environment.
Administration Server 	One instance of WebLogic Server in each domain acts as an Administration Server, providing a central point for managing all server instances.
Managed Server 	Hosts application components and resources in a domain. A machine may host multiple Managed Servers.
Node Manager 	A Java program provided with WebLogic Server that enables you to start, stop, and monitor remote WebLogic Server instances. To enable these capabilities, you run an instance of Node Manager on each physical machine in your domain.

Table 2-1 Components of a Basic Application Deployment Environment

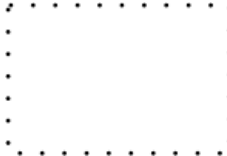


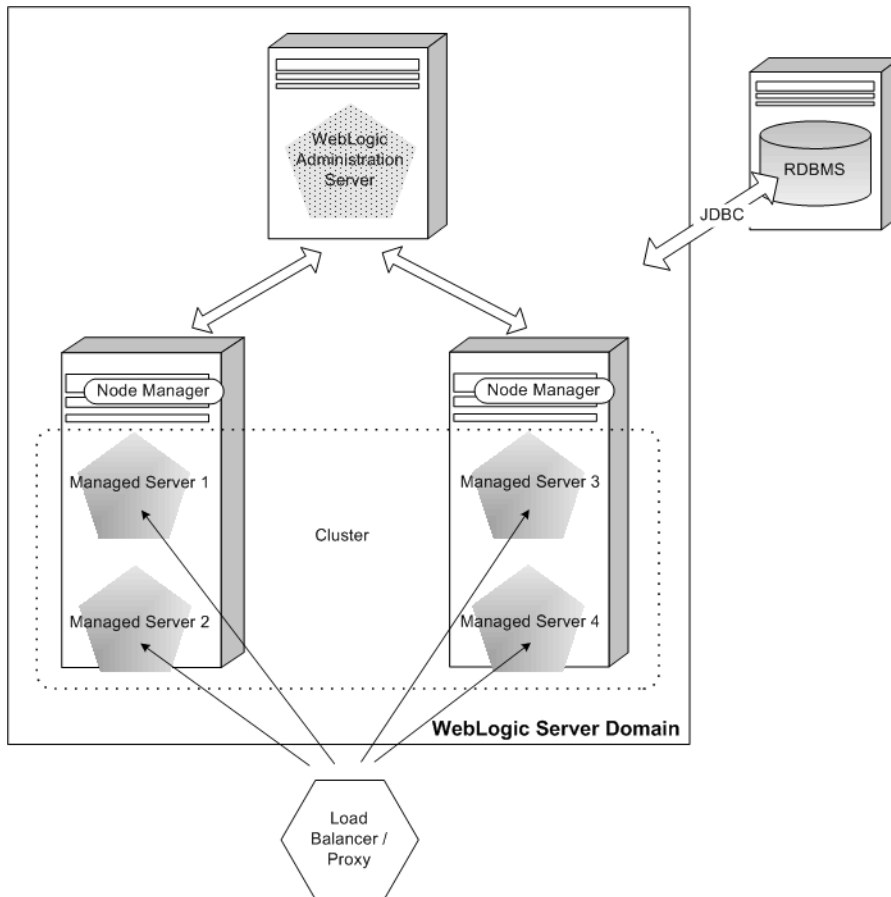
Component	Description
<p>Cluster</p> 	<p>Consists of multiple server instances running simultaneously and working together to provide increased scalability and reliability. A cluster appears to clients as a single WebLogic Server instance. The server instances that constitute a cluster can run on the same machine, or be located on different machines.</p> <p>Note: The Administration Server can never be a member of a cluster.</p>
<p>Database</p> 	<p>Hosts application and system data. WebLogic Platform supports common RDBMS systems for databases, such as PointBase, Oracle, Microsoft SQL Server, DB2, and Sybase. Interaction with an RDBMS is enabled via WebLogic JDBC.</p>
<p>Load Balancer/Proxy</p> 	<p>Distributes client connection requests, provides load balancing and failover across the cluster, and provides security by concealing the local area network addresses from external users. A load balancer/proxy can be implemented via one of the following ways:</p> <ul style="list-style-type: none"> • An instance of WebLogic Server with the <code>HttpClusterServlet</code> • A Web server supported by WebLogic Platform that is configured with the WebLogic Server proxy plug-in • A hardware load balancer

Figure 2-1 shows an example of a typical WebLogic Server domain with Managed Servers and a cluster.

Figure 2-1 Typical WebLogic Domain with Managed Servers and a Cluster



For detailed information about WebLogic Server domains, see “Overview of WebLogic Server Domains” in *Configuring and Managing WebLogic Server* at the following URL:

http://e-docs.bea.com/wls/docs81/adminguide/overview_domain.html

The section that follows discusses some of the considerations of and requirements for configuring a WebLogic Platform domain.

Considerations for Configuring WebLogic Platform Domains

When configuring a domain for WebLogic Platform applications, and that domain has at least one cluster, you should take into account the following considerations:

- The domain configuration templates that you use to create domains, namely via the Configuration Wizard or the WebLogic Scripting Tool (WLST) Offline, dictate the structure of the servers that run in that domain. That is, all the servers in that domain are configured to contain the components specified in that template. For example, if you create a WebLogic Integration domain, every server in that domain is configured with WebLogic Integration resources and not with WebLogic Portal resources. Conversely, every server in a WebLogic Platform domain is configured with the resources required to run all the WebLogic Platform components.
- You should consider using at least one cluster in any domain that is configured for a production environment. Clusters provide the sort of high-availability, load balancing, and failover features that are usually necessary for a production application.
- A WebLogic Portal application, as well as the Administration Portal tools, is usually targeted to both the cluster and the Administration Server. This is necessary to enable a very powerful feature of WebLogic Portal, which is to dynamically propagate updates and other customizations to each server instance in the domain.
- A WebLogic Integration application is targeted to the cluster, and the WebLogic Integration Administration Console is targeted to the Administration Server.
- A domain may have no more than one WebLogic Integration cluster. That is, a WebLogic Integration application, and the system-level resources required for that application, can be targeted to no more than one cluster. (Note that this restriction complicates creating a multi-cluster domain that is configured to run a WebLogic Integration application on one cluster, and a WebLogic Portal application on another cluster, discussed in [“Multi-Cluster Platform Domain Example” on page 2-17.](#))
- A WebLogic Platform domain requires a Relational Database Management System (RDBMS).
 - An RDBMS is required for the WebLogic Workshop run-time framework, and also for system- and application-level WebLogic Portal and WebLogic Integration resources.
 - A set of JDBC resources—including connection pools, data sources, and transactional data sources—is required by WebLogic Platform components for handling system-level RDBMS operations. These resources are explained in more detail in [“Considerations for Configuring and Targeting Resources” on page 3-3.](#)
- Each WebLogic Server instance in a WebLogic Platform domain is configured with a JMS server. A JMS server manages all the destinations, connection factories, message stores, keys, etc. required for both the application as well as system-level resources that use JMS.

In addition, each Web application project that you create in WebLogic Workshop that makes asynchronous requests to Web services, including for WebLogic Integration and WebLogic Portal, by default has an asynchronous queue and an asynchronous error queue for which you must create a pair of corresponding JMS queues on each target server on which the application is deployed. If the target domain is clustered, you must configure these queues as distributed queues. For more information, see [“Adding Application Resources Required by the WebLogic Workshop Runtime”](#) on page 3-13.

- A cluster should have a minimum of three Managed Servers for load balancing. In the event that one Managed Server fails, load balancing can continue between the two remaining Managed Servers.

WebLogic Platform Domain Examples

This section describes considerations for the configuration of WebLogic Platform applications in the following four example domain environments:

- [Development Domain Example](#)
- [Single-Cluster Platform Domain Example](#)
- [Multi-Domain Example](#)
- [Multi-Cluster Platform Domain Example](#)

The purpose of presenting these examples is two-fold:

1. To describe WebLogic Platform configurations that can be used as stages in the promotion process.
2. To introduce the scope of the deployment and configuration tasks that are described later in this document.

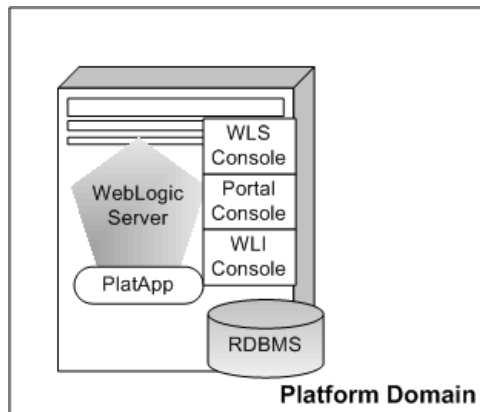
These examples are not meant to imply any limitations on the variety of configurations that are possible with WebLogic Platform. There are other configurations that work equally well; however, if you are new to WebLogic Platform, these examples are a useful starting place.

In the diagrams shown in these sections, the types of applications running on a machine imply the minimum set of WebLogic Platform products installed on that machine. For example, a machine hosting a WebLogic Portal application must have WebLogic Portal, WebLogic Workshop, and WebLogic Server installed. However, the types of applications running on a given machine do not imply that only those products are installed. For example, it is possible that a WebLogic Portal application can run on a machine on which all WebLogic Platform products, including WebLogic Integration, are installed.

Development Domain Example

A typical development environment for using WebLogic Workshop consists of a single-server domain as shown in the following figure.

Figure 2-2 Single-Server Development Domain Example



Components Shown in Example

The development environment shown in [Figure 2-2](#) has the following characteristics:

- The entire domain, including the database, is located on one machine.
- The RDBMS, which manages the database required by system-level components of WebLogic Platform, may be the default single-user PointBase RDBMS. Or, it can be any production-capable RDBMS supported by WebLogic Platform. Note that it is a best practice to use the same RDBMS in development that will be used in the production system; however, this is not a requirement. WebLogic Platform provides scripts that simplify switching from one database and RDBMS to another, which include extracting the database schemas from one database and inserting them into another.
- The domain in this example includes a single server that hosts the single WebLogic Platform application, shown as **PlatApp**, in this figure. As defined in [“About WebLogic Platform Applications” on page 1-1](#), a WebLogic Platform application incorporates all components of WebLogic Platform.
- The server also hosts all the administration components of WebLogic Platform, which are the following:

- The WebLogic Server domain administration components, represented as **WLS Console**
- WebLogic Portal system-level Web applications and EJBs, represented as **Portal Console**. These components include the WebLogic Administration Portal and the datasync Web application.
- WebLogic Integration Administration Console and other administrative components, represented as **WLI Console**. These components include EJBs and Web applications, startup and shutdown classes, and configuration queues and topics.
- WebLogic Platform domain JMS and JDBC resources.

Considerations for this Configuration

This is a basic WebLogic Platform domain configuration and is typically used for application development. Server administration is simplified, and all application-level and system-level resources are automatically configured and located on the one server. Considerations for using this configuration for development include the following:

- This domain is created with the Configuration Wizard using the Express mode, which is the easiest and quickest way to create a domain and is ideal for a development environment. Express mode uses the default settings from the configuration template. Keep in mind that in this mode you cannot modify template settings (for example, server port numbers) while you are creating your domain.
- The server in this domain is configured with all the resources necessary to run any kind of application supported in WebLogic Platform; that is, WebLogic Server, WebLogic Workshop, WebLogic Integration, and WebLogic Portal.
- This domain is configured to run in development mode, which is different from production mode in the following ways:
 - The use of demo security certificates is enabled
 - WebLogic Server instances automatically deploy and update applications
 - A limited number of system-level resources are made available by default

For more information about how resources are configured in a domain created in development mode, see “Differences Between Configuration Startup Modes” in “Creating a New WebLogic Domain” in *Creating WebLogic Configurations Using the Configuration Wizard* at the following URL:

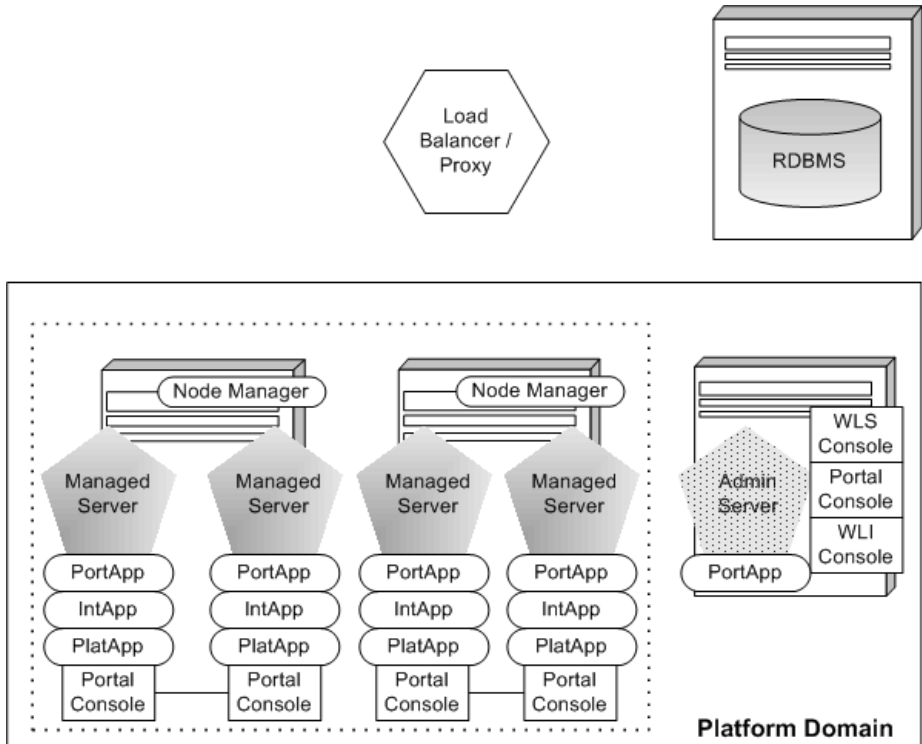
<http://e-docs.bea.com/platform/docs81/configwiz/newdom.html>

- All security data is stored in the WebLogic embedded LDAP server. Access to the domain is trusted by default.
- This configuration is not recommended for a production environment. Because all system and application components are on a single machine, if the machine fails, the entire system fails.

Single-Cluster Platform Domain Example

A single-cluster WebLogic Platform domain environment is shown in the following figure.

Figure 2-3 Single-Cluster Platform Domain Example



Components Shown in Example

The production environment shown in [Figure 2-3](#) has the following characteristics:

- The domain in this configuration is based on the Basic WebLogic Platform Domain template.
- This domain has a single cluster, which is composed of four Managed Servers:
 - Two Managed servers are assigned to one machine, and two are assigned to a second machine.
 - Each machine in the cluster has WebLogic Platform installed. Each machine is also configured with a Node Manager.
- Each managed server instance hosts three applications:
 - A WebLogic Portal application, represented as **PortApp**
 - A WebLogic Integration application, represented as **IntApp**
 - A WebLogic Platform application, represented as **PlatApp**.

Each of these applications is deployed as an EAR file; that is, each is a discrete deployment unit.

- Each managed server instance also hosts the WebLogic Administration Portal, WebLogic Portal administration tools, and the datasync Web application. The collection of these system-level resources is represented as **Portal Console**.
- The Administration Server is assigned to a third machine, and hosts the following:
 - WebLogic Server domain administration infrastructure, which includes the WebLogic Server Administration Console, represented as **WLS Console**.
 - The WebLogic Administration Portal, WebLogic Portal administration tools, and the datasync Web application, represented as **Portal Console**.
 - WebLogic Integration Administration Console, and other WebLogic Integration administrative EJBs and Web applications, startup and shutdown classes, and configuration queues and topics, represented as **WLI Console**.
 - The WebLogic Portal application is also typically deployed on the Administration Server, represented as **PortApp**, but is used for the purposes of propagating WebLogic Portal data throughout the domain, and not for handling client requests.
- A load balancer/proxy server routes client requests to an available server in the cluster. Using this component is a requirement for handling client requests to applications that are deployed on a cluster, but it also implements important practices for a production system:
 - Applications on the cluster are accessed externally by the IP address of the machine on which the load balancer/proxy is hosted. Therefore, client applications never directly

address the specific machine on which the application is hosted, which is very desirable from a security standpoint.

- You can use the load balancer/proxy to prevent external access to the Administration Server, which provides increased security.
 - A load balancer/proxy allows you to have in-memory replication of client session information, which is very desirable from an availability standpoint. In the event of the failure of a machine on which a server is hosted, the client session can be reestablished with another running server instance.
 - You can add Managed Servers to the cluster without affecting how clients access the applications hosted on those servers.
- The RDBMS is hosted on a standalone machine.

Considerations for this Configuration

By incorporating multiple machines and servers into a domain, it becomes possible to enable the sort of availability and reliability capabilities in WebLogic Platform that are necessary in a production system. For example, by configuring a cluster, multiple servers can perform similar functions. Any server in the cluster can process a request from a particular client. In addition, a cluster automatically provides load balancing across servers.

Note the following considerations about this configuration:

- This domain can be created using either the Configuration Wizard or the WebLogic Scripting Tool (WLST) Offline. When you create a domain similar to the one in this example, you cannot use Express mode in the Configuration Wizard; instead, you need to use Custom mode so that you can configure the cluster and other resources required for this configuration.

An example of creating a single-cluster production domain using WLST Offline is provided in [“Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline” on page 3-21](#).

- This single-domain configuration is simplest for a WebLogic Platform application in a production environment. Controls can be leveraged in WebLogic Workshop to simplify application development and intra-application requests and responses. Having portals, business processes, etc. in the same EAR file simplifies deployment. In addition, several individual components of WebLogic Platform include sample applications that demonstrate best practices of design and development for single-cluster environments.

However, this configuration requires targeting of individual application modules so that the WebLogic Integration Web application and EJBs are targeted only to the cluster, and

WebLogic Portal EJBs and Administration Portal are targeted to both the Administration Server and cluster.

For information about targeting considerations of application modules in a WebLogic Platform application, see [“Application Targets” on page 8-3](#).

- The Administration Server should not handle client application requests. Therefore, in general, applications should only be targeted to the cluster. This way, the Administration Server is never exposed externally to clients of the applications deployed in the domain. One exception to this is WebLogic Portal applications, discussed next.

Also, note that the Administration Server must be running at all times to support WebLogic Platform applications.

- WebLogic Portal applications are deployed on the Administration Server as well as the cluster so that customizations and other dynamic updates to the portal application can be propagated to all deployed instances of the application throughout the domain. However, only those instances of the application that are deployed on the cluster can be accessed by external clients, which is done via the load balancer/proxy.

The files in a WebLogic Portal application include several system-level EJBs, JAR files, Web applications, and Web services, such as the following:

- WebLogic Administration Portal
- Datasync Web application
- Content management
- Personalization
- Pipeline services or commerce services, if installed

These components are copied automatically into the application project directory tree by the WebLogic Workshop IDE when you create the application. You typically deploy the EAR file on all servers in a domain; however, it is possible to target only the WebLogic Portal application to the cluster, and the Administration Portal and other administrative components only on the Administration Server.

- As noted in [“Considerations for Configuring WebLogic Platform Domains” on page 2-4](#), each Web application project that you create in WebLogic Workshop that makes asynchronous requests to Web services has an asynchronous queue and an asynchronous error queue for which you must create a pair of corresponding distributed JMS queues on the cluster. The names of these queues are identified in the `wlw-manifest.xml` file that WebLogic Workshop provides with each EAR file. For information about creating these

distributed queues, see [“Adding Application Resources Required by the WebLogic Workshop Runtime”](#) on page 3-13.

- In a production environment, the domain should be configured with a database managed by an enterprise-level RDBMS supported by WebLogic Platform. This database is not located any machine on which WebLogic Platform, or any of its components, is installed.

There are a number of steps you need to take to configure the database for the target environment, including:

- Creating and preparing the database
- Promoting WebLogic Integration and WebLogic Portal database schema

Configuring the database is discussed in [Chapter 4, “Configuring the Production Database.”](#) This chapter includes an example of using WLST to load WebLogic Portal and WebLogic Integration database schema.

- Security considerations include the following:
 - In a production environment, insulate the domain infrastructure from potential attacks by always locating the Administration Server on a separate machine from those that host Managed Servers, and do not include the Administration Server in any cluster. As stated previously, do not target applications on the Administration Server (with the exception of WebLogic Portal applications, as noted).
 - By default, user and group information is stored in the embedded LDAP server, which is fully adequate for development, system integration, testing, and other pre-production domain environments. The WebLogic Security Service makes it easy to export security information, such as policies set on domain resources, including applications, and the configuration of security providers, from one deployment environment and import it into another. For more information, see [“Promoting Embedded LDAP Security Data to the Target Database”](#) on page 5-7.

However, in a production environment within an enterprise, user and group information is typically stored in the LDAP server used by the enterprise. Therefore, a key step in promoting an application into a production environment is configuring security providers with that external LDAP server. For information about using an external LDAP server, see [“Using an External Store for User Information”](#) in *Security in WebLogic Platform 8.1*.

- During domain creation, there is dynamic table creation for WebLogic Server subsystems, such as JMS when JMS is configured to use a JDBC store. Administrators need the appropriate access granted to the database so that they can update the tables as

necessary. Therefore, when configuring the domain, make sure that the appropriate database access is granted.

- Note that WebLogic Platform supports the use of multiple authentication providers. For information about configuring multiple authentication providers with WebLogic Server, see [“Configuring Security Providers”](#) in *Managing WebLogic Security*. For information about managing multiple authentication providers in the WebLogic Administration Portal, see [Using Multiple Authentication Providers with WebLogic Portal](#). For information about using multiple authentication providers in WebLogic Portal application development, see [Using Multiple Authentication Providers in Portal Development](#).
- When configuring a production domain, you might want to configure each Managed Server for SSL so that only SSL-based requests are received by the applications deployed on those servers. However, depending on your environment, it might be preferable to enable SSL in the load balancer, or to enable SSL at the application level and not for the entire Managed Server, to save on the overhead of enabling SSL by default for the entire server.

For information about how to modify your applications’ deployment descriptors so that they receive SSL-based requests only, or allow only authenticated users to have access, see [“Preparing Application Security”](#) on page 7-16.

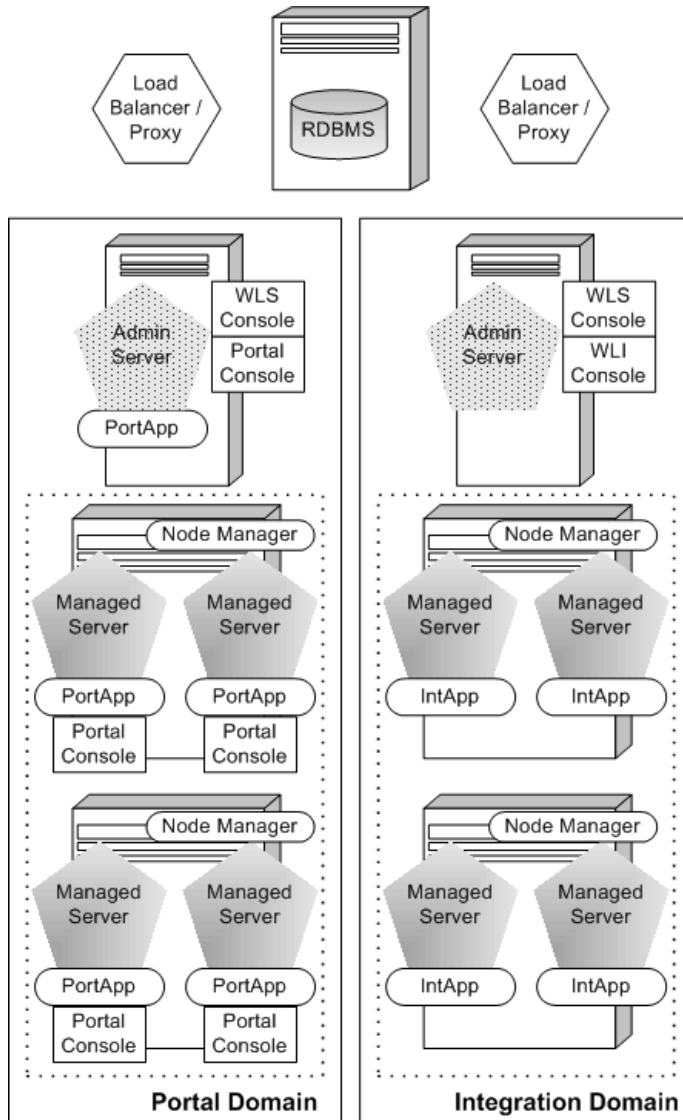
- The use of a load balancer/proxy introduces considerations for session replication. For information about configuring a load balancer/proxy, see [Chapter 6, “Using Load Balancers and Web Proxy Servers.”](#)

If you are promoting an application from a nonclustered to a clustered environment, there are considerations for accessing other applications. Instead of accessing other applications directly, clients must access it via a load balancer/proxy. [“Enabling Inter-Application Communication”](#) on page 7-11 explains how to modify application files so that access via a load balancer is enabled.

Multi-Domain Example

A loosely coupled, multi-domain production environment is shown in the following figure.

Figure 2-4 Multi-Domain Example



Components Shown in Example

The production system shown in [Figure 2-4](#) is characterized by the following:

- A WebLogic Portal domain is created based on the Basic WebLogic Portal Domain template. This domain contains:
 - A cluster hosted on two machines
 - Each machine in the cluster hosts two Managed Servers
 - Each Managed Server hosts one WebLogic Portal application
 - An Administration Server, which hosts the WebLogic Portal application, the Administration Portal, the datasync Web application, and WebLogic Portal system EJBs and Web services.

Note that each Managed Server, as well as the Administration Server, can host multiple WebLogic Portal applications. The Administration Server is not constrained to host only one WebLogic Portal application.

- A WebLogic Integration domain is created based on the Basic WebLogic Integration Domain template. This domain contains:
 - A cluster hosted on two machines.
 - Each machine in the cluster hosts two Managed Servers.
 - Each Managed Server hosts one WebLogic Integration application.
 - An Administration Server, which hosts the WebLogic Integration administrative Web application, WebLogic Integration system EJBs, and other system components.
- Each cluster has a load balancer/proxy for handling requests.
- A database is shared by both domains.

Considerations for this Configuration

The considerations for this multi-domain production system include the following:

- This configuration is appropriate when the WebLogic Portal domain is used as the access point to the applications and resources in the WebLogic Integration domain. This loosely coupled configuration allows individual applications to be scaled as needed:
 - A portal can be scaled broadly, independently of the WebLogic Integration application usage requirements. For example, the optimum performance might be achieved via scaling up a number of small machines on which it is deployed.
 - WebLogic Integration application might require fewer, but more powerful machines.

An example of creating this domain using WLST Offline is provided in [“Example: How to Configure a Multi-Domain Environment Using WLST Offline”](#) on page 3-28.

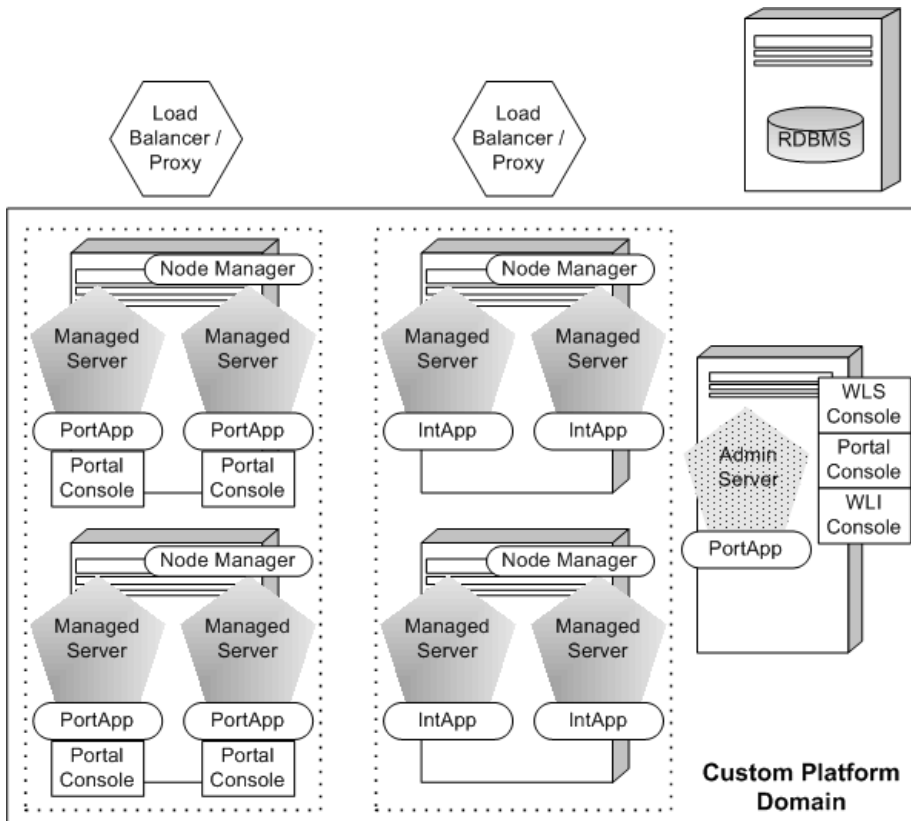
An example of creating this domain using the Configuration Wizard in silent mode is provided in [“Example: How to Configure a Multi-Domain Environment Using the Configuration Wizard in Silent Mode”](#) on page 3-29.

- This configuration separates the application types more distinctly, resulting in simplified development, configuration, and deployment tasks. However, because the applications are deployed in separate domains, this configuration results in more complex interactions between the WebLogic Portal and WebLogic Integration components: for example, a JPD Proxy or Web service interface for invoking business processes must be used.
- The security considerations described for the single-cluster example in [“Considerations for this Configuration”](#) on page 2-11 also apply to the multi-domain example. In a production environment, both domains should be configured with the user information in a central LDAP server. However, if user information is maintained in the embedded LDAP server in one of the domains, identity propagation and credential passing between the domains is more complex.
- Sharing a single database is not problematic because WebLogic Portal and WebLogic Integration do not share database schema.
- For interdomain communication, note the following considerations:
 - Trust should be enabled for RMI traffic between the two domains.
 - Consider enabling two-way SSL if your applications use Web services.
 - Use JMS for asynchronous communication, such as via a Messaging Bridge or Foreign JMS Server.
- Domain configuration and application deployment is simplified, but each domain must be managed separately.

Multi-Cluster Platform Domain Example

It is possible to create a multi-cluster domain in which WebLogic Integration and WebLogic Portal applications are deployed on separate clusters in the same domain. This configuration is shown in the following figure.

Figure 2-5 Multi-Cluster Platform Domain Example



Components Shown in Example

The production system shown in [Figure 2-5](#) is characterized by the following:

- Two clusters configured in the domain: one that has a WebLogic Integration application targeted to it, the other that has a WebLogic Portal application targeted to it.
 - Each cluster comprises two machines. Each machine is configured with a Node Manager.
 - Each machine in a cluster hosts two Managed Servers.
 - Each Managed Server in each cluster hosts one application.

- The domain shown in this example is a custom WebLogic Platform domain. Because of the restriction that limits a domain to having one and only one WebLogic Integration cluster, not all the Managed Servers in this domain are the same: the servers in the portal cluster are not configured with the resources necessary to run WebLogic Integration applications.
- This domain has a machine hosting the Administration Server. The Administration Server has the following targets:
 - WebLogic Server domain administration infrastructure.
 - The WebLogic Portal application, the Administration Portal, the datasync Web application, and WebLogic Portal system EJBs and Web services.
 - WebLogic Integration administrative EJBs and Web applications, startup and shutdown classes, and configuration queues and topics.

Configuration Considerations

Considerations for configuring a multi-cluster domain include the following:

- As with [“Multi-Domain Example” on page 2-14](#), separating the WebLogic Integration cluster from the WebLogic Portal cluster allows each application to be scaled independently.
- A single domain configuration allows:
 - Centralized management of applications and resources
 - Simplified messaging between applications; this does not require a Messaging Bridge or Foreign JMS Server, as with [“Multi-Domain Example” on page 2-14](#).
- The advantages of this configuration include:
 - You can consolidate the WebLogic Portal application cluster, enabling you to scale it independently from the WebLogic Integration application. This allows you to improve load balancing and reliability because one application continues to work if the other application becomes unavailable or intermittently available.
 - The machines hosting the Managed Servers do not require as much memory because the footprint of the application, and the WebLogic Platform resources that support it, is smaller. This allows you to more finely tune the functions each server is performing.
- Creating a multi-cluster domain that includes a WebLogic Integration application is complex, but it can be created in multiple ways. [“Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline” on page 3-30](#) provides an example of creating this configuration. Some of the steps can be automated via WLST Offline, but

some steps need to be performed manually. The following summarizes the steps used in this example:

- a. Using WLST Offline, the domain is created using the Basic WebLogic Platform Domain template.
 - b. Two clusters are configured: one for WebLogic Integration and the other for WebLogic Portal.
 - c. Four machines are configured: two are assigned to the WebLogic Integration cluster, and two to the WebLogic Portal cluster.
 - d. The JMS JDBC store is configured for the WebLogic Portal cluster.
 - e. The targets for the system resources and WebLogic services for each cluster are manually reassigned: WebLogic Portal resources are unassigned from the WebLogic Integration cluster, and WebLogic Integration resources are unassigned from the WebLogic Portal cluster.
 - f. The targets for the applications are corrected. First, all applications are unassigned from the Administration Server, the first Managed Server assigned in the WebLogic Integration cluster, and both clusters. Next, each application is assigned to its correct targets.
 - g. The domain `config.xml` file is modified manually to create JMS resources for the WebLogic Portal cluster.
- As with the preceding single-domain and multi-domain configuration examples, you need to create resources based on the content of the `wlw-manifest.xml` file. But you need to distinguish between the resources that must be created on each cluster.
 - This configuration requires a load balancer/proxy for each cluster.
 - The security considerations described for the previous two configuration examples apply to the multi-cluster configuration. User information can be stored in the embedded LDAP server or in an external LDAP server. Both clusters can use the domain security configuration easily.
 - Communication between the clusters is implemented via messaging; for example, a JMS reliable transport, MQ Series when you need a Messaging Bridge, or calls from a control. This configuration requires precise targeting of applications and also of system-level JMS and JDBC resources. Consider enabling two-way SSL if your applications use Web services.

Creating and Configuring the WebLogic Domain

A domain is the basic administration unit for WebLogic Server. To run WebLogic Platform applications, you must create a domain that includes the appropriate WebLogic Platform resources. For more detailed information about WebLogic Server domains, see [“Overview of WebLogic Server Domains”](#) in *Configuring and Managing WebLogic Server*.

This section lists the tools that you can use to configure a WebLogic domain; provides considerations for configuring WebLogic resources, clusters, and targets; and describes how to set up and start the servers.

Topics include:

- [“Tools for Configuring the Target Domain”](#) on page 3-2
- [“Considerations for Configuring and Targeting Resources”](#) on page 3-3
- [“Adding Application Resources Required by the WebLogic Workshop Runtime”](#) on page 3-13
- [“Configuring Servers to Start in Production Mode”](#) on page 3-14
- [“Setting the SDK”](#) on page 3-14
- [“Setting Up the Managed Servers on Remote Machines”](#) on page 3-14

The following examples illustrate how to configure a WebLogic domain using scripts. Scripts facilitate controlled and repeatable creation of domains in different target environments.

- [“Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline”](#) on page 3-21

- “Example: How to Configure a Multi-Domain Environment Using WLST Offline” on page 3-28
- “Example: How to Configure a Multi-Domain Environment Using the Configuration Wizard in Silent Mode” on page 3-29
- “Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline” on page 3-30

Note: Before proceeding, review your promotion plan to understand the configuration and targeting requirements of each resource required for your application. For more information about promotion plans, see “Planning the Promotion” on page 1-5.

Tools for Configuring the Target Domain

The tools used to configure the target domain depend on whether you are creating a new target domain or the domain already exists. The tools are summarized in the following table.

Table 3-1 Tools for Configuring the WebLogic Domain

To...	Do the following...
Create a new WebLogic domain	<p>Create the WebLogic domain and define the required WebLogic resources using the Configuration Wizard or WebLogic Scripting Tool (WLST) Offline. A set of predefined configuration templates is provided to help get you started.</p> <p>The Configuration Wizard or WLST Offline configures many of the resources for you, as explained in the following sections.</p> <p>For more information about creating the WebLogic domain using:</p> <ul style="list-style-type: none">• Configuration Wizard, see “Creating a New WebLogic Domain” in <i>Creating WebLogic Configurations Using the Configuration Wizard</i>• WLST Offline, see the instructions for setting up and using WLST Offline, as well as sample scripts for configuring domains, at the following URL: https://codesamples.projects.dev2dev.bea.com/servlets/Scarb?id=97 <p>After you create a new WebLogic domain, you can update it using one of the methods described below in Update an existing WebLogic domain.</p>

Table 3-1 Tools for Configuring the WebLogic Domain (Continued)

To...	Do the following...
Update an existing WebLogic domain	Update an existing WebLogic domain using one of the following tools: <ul style="list-style-type: none"> • WebLogic Server Administration Console (or the equivalent command-line interface), as described in <i>Configuring and Managing WebLogic Server</i> • Configuration Wizard, as described in “Extending Domains” in <i>Creating WebLogic Configurations Using the Configuration Wizard</i> • WLST Offline, as described in the WLST Offline help kit at the following URL: https://codesamples.projects.dev2dev.bea.com/servlets/Scarb?id=97

Considerations for Configuring and Targeting Resources

When creating a WebLogic domain, you need to configure and specify targets for the system resources. [Table 3-2](#) provides tips for configuring and targeting resources in a production environment.

Note: The Configuration Wizard and WLST Offline have an autoconfiguration feature that streamlines the process of configuring resources in multi-server (e.g., clustered) domains. See “[Autoconfiguration Using the Configuration Wizard and WLST Offline](#)” on page 3-12.

Table 3-2 Considerations for Configuring and Targeting Resources

Resource	Configuration Considerations	Targeting Considerations
Administration Server	<ul style="list-style-type: none"> • Set the name, address, and ports for the Administration Server using the guidelines specified in “Guidelines for Specifying the Names and Addresses of the Members in a Cluster” on page 3-12. • Enable the Secure Sockets Layer (SSL) protocol if you wish to enable secure communication between applications connected through the Web. For additional information about security considerations, see “Configuring Security” on page 5-1. • Enable HTTP tunneling to enable a stateful connection by simulating a T3Connection. For more information, see “Setting Up WebLogic Server for HTTP Tunneling” in “Configuring Web Server Functionality for WebLogic Server” in <i>Configuring and Managing WebLogic Server</i>. <p>Related topics:</p> <ul style="list-style-type: none"> • For an example using WLST Offline, see “Configure the Administration Server” on page 3-23. 	<p>Ensure that the Administration Server IP address is not included in the cluster-wide DNS name. You should not include the Administration Server in the cluster for the following reasons:</p> <ul style="list-style-type: none"> • There is no benefit to doing so. The administrative objects are not clusterable, and will not fail over to another cluster member if the Administration Server fails. • The Administration Server should not be configured to handle client requests.

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
Managed Servers	<ul style="list-style-type: none"> • Set the name, address, and ports for each Managed Server using the guidelines specified in “Guidelines for Specifying the Names and Addresses of the Members in a Cluster” on page 3-12. • When using the WebLogic Server as a Web proxy server, add a Managed Server to function as the proxy. For more information see “Load Balancing with a Web Proxy Server” on page 6-2. • Increase the thread count value based on your performance requirements. See “Tuning the Default Execute Queue Threads” in <i>WebLogic Server Performance and Tuning</i>. <p>Related topics:</p> <ul style="list-style-type: none"> • For an example using WLST Offline, see “Configure the Cluster, Managed Servers, and Machines” on page 3-25. 	<ul style="list-style-type: none"> • If you added a Managed Server to act as a proxy server, do not include it in the cluster. • Avoid deploying server instances in a cluster across a firewall. For more information, see “Firewalls Can Break Multicast Communication” in “One-to-Many Communication Using IP Multicast” in “WebLogic Server Communication in a Cluster” in “Communications in a Cluster” in <i>Using WebLogic Server Clusters</i>.

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
Cluster	<ul style="list-style-type: none"> • Identify the address and port for multicast communications in the cluster. Multiple clusters on a network may share a multicast address and multicast port combination if necessary. Ensure that the multicast address for a cluster is not used for any purpose other than cluster communications. • Set the name, address, and ports for each member in the cluster using the guidelines specified in “Guidelines for Specifying the Names and Addresses of the Members in a Cluster” on page 3-12. • Failover for entity beans and EJB handles requires that the cluster address be specified as a DNS name that maps to all server instances in the cluster and only server instances in the cluster. For more information, see “Failover for Entity Beans and EJB Handles” in “Replication and Failover for EJBs and RMI” in “Failover and Replication in a Cluster” in <i>Using WebLogic Server Clusters</i>. <p>Related topics:</p> <ul style="list-style-type: none"> • Using WebLogic Server Clusters • “Load Balancing in a Cluster” and “Failover and Replication in a Cluster” in <i>Using WebLogic Server Clusters</i> • For an example using WLST Offline, see “Configure the Cluster, Managed Servers, and Machines” on page 3-25. 	<p>To review considerations for targeting applications to the cluster, see “Application Targets” on page 8-3.</p>

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
Machines	<p>Define machine names if:</p> <ul style="list-style-type: none"> • You are running Node Manager on a machine that does not also host the Administration Server (e.g., a remote machine). In this case, do not set the Node Manager to <code>localhost</code>. If you identify the Listen Address by IP address, you must disable <code>HostNameVerification</code> on Administration Servers and Managed Servers that use Node Manager. • You are running more than one server instance per machine so that WebLogic Server does not replicate a session on the same machine. <p>Related topics:</p> <ul style="list-style-type: none"> • For an example using WLST Offline, see “Configure the Cluster, Managed Servers, and Machines” on page 3-25. 	N/A

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
JDBC Connection Pools	<p>For each JDBC connection pool, set the database type and related information, such as the host and port information, to specify the <i>production</i> database. For a description of the predefined JDBC configuration information for the WebLogic Platform domain, see “Deployment Targeting Reference” on page A-1.</p> <p>Please note the following:</p> <ul style="list-style-type: none"> • Configure all default connection pools (such as <code>cgPool</code>, <code>cgJMSPool-nonXA</code>, <code>bpmArchPool</code>, and so on). • If you change the name of a connection pool, target the existing data sources to the new connection pool. • Configure the <code>cgJMSPool-nonXA</code> pool with a nonXA driver. • If the reporting data tables are referenced through a different pool than <code>cgPool</code>, then when configuring <code>bpmArchPool</code>, configure it as the Reporting Datastore in the WebLogic Administration Console. For more information, see “Configuring the Reporting DataStore” in “System Configuration” in <i>Managing WebLogic Integration Solutions</i>. • When creating XA domains to support global transactions, review the JDBC configuration guidelines in “Creating XA Domains Using Configuration Templates” in <i>Creating WebLogic Configurations Using the Configuration Wizard</i>. <p>Related topics:</p> <ul style="list-style-type: none"> • For a list of supported database drivers, see: “Supported Database Configurations” in <i>Supported Configurations for WebLogic Platform 8.1</i>. • “Load Balancing in a Cluster” and “Failover and Replication in a Cluster” in <i>Using WebLogic Server Clusters</i> • For an example using WLST Offline, see “Configure the JDBC Connection Pools” on page 3-24. 	<ul style="list-style-type: none"> • Target to the cluster; you can target the same JDBC connection pool to multiple clusters • Target to the Administration Server, if required (for example, if a JMS Server is targeted to the Administration Server) • Ensure that you target to the same servers and clusters as the related JDBC data sources

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
JDBC Data Source/ TX Data Source	<p>Configure one JMS JDBC Store for each server that has a JMS Server defined.</p> <p>Please note the following:</p> <ul style="list-style-type: none"> • If you are using an XA driver, configure only one Tx Data Source per XA-compliant JDBC connection pool. Note that you can specify multiple JNDI names per Tx Data Source, separated by semicolons. • If you are using distributed transactions with non-XA drivers, configure all Tx Data Sources on the same connection pool. • If you are using only local transactions, use a Data Source. Otherwise, use a Tx Data Source, but ensure that you suspend and subsequently resume any global transaction currently associated with your thread around your local transaction boundary. • If you create multiple domains that share the same database, each domain must reference unique database schemas. • You should not edit the JNDI name of the JDBC data sources or Tx data sources. • When creating XA domains to support global transactions, review the JDBC configuration guidelines in “Creating XA Domains Using Configuration Templates” in “How Do I...?” in <i>Creating WebLogic Configurations Using the Configuration Wizard</i>. • When creating XA domains to support Oracle RAC, review the JDBC configuration guidelines in “How Do I: Create XA domains with MultiPools and an Oracle RAC Database?” in “How Do I...?” in <i>Creating WebLogic Configurations Using the Configuration Wizard</i>. <p>Related topics:</p> <ul style="list-style-type: none"> • For a description of the predefined JDBC configuration information for the WebLogic Platform domain, see “Deployment Targeting Reference” on page A-1. 	<ul style="list-style-type: none"> • Target the same servers and clusters as the related JDBC connection pool • You can target the same JDBC data source to multiple clusters

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
JDBC MultiPools	<p>Create JDBC MultiPools to provide additional load balancing support. MultiPools are not required to be used in a cluster.</p>	<ul style="list-style-type: none"> • Target to the cluster; you can target the same JDBC MultiPool to multiple clusters • Target to the Administration Server, if required • Ensure that you target to the same servers and clusters as the related JDBC data sources and connection pools
JMS Servers	<p>To support JMS failover, target JMS Servers to migratable Managed Servers. This targeting is done automatically, when using the Configuration Wizard or WLST Offline to create the domain.</p> <p>WebLogic JMS takes advantage of the migration framework implemented in the WebLogic Server core for clustered environments by configuring migratable targets for JMS Servers. This framework enables JMS to properly respond to migration requests and bring a JMS server online and offline in an orderly fashion. If you do not configure a migratable target in a cluster, you can still migrate migratable services manually to any server instance in the cluster. For more information, see “Configuring Migratable JMS Server Targets” in “Managing WebLogic JMS” in <i>Programming WebLogic JMS</i>.</p> <p>If you add a JMS server when creating a domain, autoconfiguration is not performed. For more information about autoconfiguration, see “Autoconfiguration Using the Configuration Wizard and WLST Offline” on page 3-12.</p> <p>Note: WebLogic Integration system queues can be mixed with application queues on the same JMS server.</p>	<ul style="list-style-type: none"> • Target one JMS Server per Managed Server, as required • Target to the Administration Server, as required (for example, to support WebLogic Portal applications)

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
JMS Destinations/ Distributed Destinations	<ul style="list-style-type: none"> • Create JMS distributed destinations when using a cluster to support high availability and load balancing of physical destinations. When using the Configuration Wizard or WLST Offline, by creating the destinations <i>before</i> you create the cluster, the destination will be configured automatically as a distributed destination with the same JNDI name, and the members will be configured as well. • Create one JMS distributed destination member for each Managed Server. • Set the <code>RedeliveryLimit</code> to a number of messages that is practical for the environment. By default, a JMS queue will redeliver error messages and warnings an infinite number of times. • Create an error destination for undeliverable messages. The error destination can be either a queue or a topic destination. It must be configured on the same JMS server as the destination for which it is defined. If no error destination is configured, then undeliverable messages are simply deleted. You should set the <code>RedeliveryLimit</code> to 0 for a JMS queue. For more information, see “JMS Destination Tasks” in the <i>Administration Console Online Help</i>. • Configure the asynchronous dispatcher queues, if necessary, as described in “Adding Application Resources Required by the WebLogic Workshop Runtime” on page 3-13. 	<p>Target to a single cluster.</p> <p>By default, the Configuration Wizard automatically assigns each JMS distributed destination to the first cluster in the domain. Once the domain is created, you can modify the target using the WebLogic Server Administration Console (or equivalent tool).</p>
	<p>Note: If you add a JMS Server and subsequently add a JMS destination to it when creating a domain, autoconfiguration is not performed. For more information about autoconfiguration, see “Autoconfiguration Using the Configuration Wizard and WLST Offline” on page 3-12.</p>	
	<p>Related topics:</p>	
	<ul style="list-style-type: none"> • “Load Balancing in a Cluster” and “Failover and Replication in a Cluster” in <i>Using WebLogic Server Clusters</i> 	

Table 3-2 Considerations for Configuring and Targeting Resources (Continued)

Resource	Configuration Considerations	Targeting Considerations
JMS File/JDBC Store	<ul style="list-style-type: none"> Define JMS file stores or JDBC stores for each of the Managed Servers and the Administration Server, as required. Configure the JMS JDBC Store with a non-XA JDBC connection pool. 	N/A

Guidelines for Specifying the Names and Addresses of the Members in a Cluster

When configuring a cluster, you specify name and address information for the cluster and its members—the server instances that comprise it—using either IP addresses or DNS names. Consider the following when configuring the names and addresses:

- Each cluster member must have a unique name.
- When specifying the listen address for each server instance in a cluster, it is recommended that you use DNS names rather than IP addresses. IP addresses can result in translation errors if you use firewalls. For more information, see “Firewall Considerations” in “Clustering Best Practices” in *Using WebLogic Server Clusters*.
- If the internal and external DNS names of a WebLogic Server instance are not identical, use the `ExternalDNSName` attribute for the server instance to define the server external DNS name. However, the `ExternalDNSName` attribute should not be used if clients are accessing WebLogic Server over the default channel and T3.

For more information on the above, see “Identify Names and Addresses” in “Before You Start” in “Setting up WebLogic Clusters” in *Using WebLogic Server Clusters*.

Autoconfiguration Using the Configuration Wizard and WLST Offline

The Configuration Wizard and WLST Offline provide an autoconfiguration feature that streamlines the process of configuring resources in multi-server (e.g., clustered) domains.

When you add Managed Servers during domain creation, the Configuration Wizard and WLST Offline will automatically distribute resources on those servers.

When you use WLST Offline to create the queues in a domain, the tool automatically configures each queue as a distributed queue and creates associated member queues on each Managed Server in the cluster. This must be done when *creating* the domain and before the cluster is created; this feature is not supported when *updating* an existing domain.

For more information about autoconfiguration, see “Autoconfiguration of Applications and Services” in “[Configuring Targets](#)” in *Creating WebLogic Configurations Using the Configuration Wizard*.

For a list of default targets for the predefined resources, see:

- “[Deployment Targeting Reference](#)” on page A-1
- “Understanding Portal Resources” in “[Configuring a Portal Cluster](#)” in the WebLogic Portal *Production Operations User Guide*
- “Deploying WebLogic Integration Resources” in “[Designing a Clustered Deployment](#)” in “[Understanding WebLogic Integration Clusters](#)” in *Deploying WebLogic Integration Solutions*

Note: Autoconfiguration for a cluster only works when you add the cluster when you *create* a domain, or when you add it when you update a domain using an *extension template*. This feature is not supported when *updating* an existing domain (without an extension template).

The Configuration Wizard and WLST Offline support autoconfiguration for single-cluster domains. When creating two clusters, you need to manually correct the targets assigned for the resources, services, and applications, as illustrated in “[Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline](#)” on page 3-30.

Adding Application Resources Required by the WebLogic Workshop Runtime

If Web services are configured to support asynchronous requests, WebLogic Workshop creates asynchronous request and error queues in the development environment to handle the requests. For each pair of queue elements, you must create a pair of corresponding JMS queues on the target server, and associate the asynchronous error queue with the asynchronous queue.

WebLogic Workshop defines the asynchronous queues in the `/META-INF/wlw-manifest.xml` file with the following element names: `<con:async-request-queue>` and `<con:async-request-error-queue>`. The queues are named using the following format: `web_app.queue.AsyncDispatcher` and `web_app.queue.AsyncDispatcher_error`, where `web_app` specifies the name of the Web application.

If the target domain is clustered, you must configure the queues as distributed queues. If you are using multiple clusters, you must create an additional set of queues for each cluster.

Note: When you use WLST Offline to create the queues in a domain, the tool automatically configures each queue as a distributed queue and creates associated member queues on each Managed Server in the cluster. This must be done when *creating* the domain and before the cluster is created; this feature is not supported when *updating* an existing domain. For more information, see [“Autoconfiguration Using the Configuration Wizard and WLST Offline” on page 3-12](#).

For an example of setting asynchronous dispatcher queues, see [“Configure the Asynchronous Dispatcher Queues Required by the Application” on page 3-23](#).

You can configure the server dispatch policy for asynchronous requests using the `<async-dispatch-policy>` element in the `wlw-config.xml` configuration file. The `<async-dispatch-policy>` element associates a project’s asynchronous requests with a server-defined dispatch policy. The dispatch policy designates which server execute thread pool the EJB should run in. The value provided for this element is used to generate an `<async-dispatch-policy>` entry in the `weblogic-ejb-jar.xml` configuration file for the `AsyncDispatcher` and `AsyncErrorDispatcher` message-driven beans. For more information, see the `<asynch-dispatch-policy>` element description in *WebLogic Workshop Online Help*.

Configuring Servers to Start in Production Mode

Set the startup mode to `prod`. For information about the differences between development and production modes, see [“Differences Between Configuration Startup Modes”](#) in *Creating WebLogic Configurations Using the Configuration Wizard*. For an example of configuring the server startup mode, see [“Set Domain Options” on page 3-27](#).

Setting the SDK

Set the SDK to one that is supported on the platform you are using, and that is optimized for production-level performance. Ensure that you configure the same SDK across all server instances in a cluster. For a list of the Java SDKs that are supported for a specific platform, see [Supported Configurations for WebLogic Platform 8.1](#). For an example of configuring the SDK see [“Set Domain Options” on page 3-27](#).

Setting Up the Managed Servers on Remote Machines

Once you have configured the target domain, you need to set up the Managed Server directories on remote machines and optionally configure Node Manager.

- [Setting Up the Managed Server Directories](#)
- [Configuring the Managed Server Start Attributes](#)
- [Configuring Node Manager](#)

Setting Up the Managed Server Directories

To set up the Managed Server directories that run on remote machines:

1. If you will not be using Node Manager to manage a remote Managed Server, create a domain on that server. Ensure that you configure the same JDK across all server instances in a cluster and that you set the mode to `prod`.

If you will be using Node Manager, a remote Managed Server directory does not require the complete set of files that are located in the Administration Server directory, as the Administration Server maintains the configuration information for all Managed Servers within a domain.

2. If you use WebLogic Workshop `wlw-runtime-config.xml` configuration file, you must copy it to the root directory of each Managed Server in the cluster.
3. Ensure the key stores for the following security services are made available to the Managed Server.

Security Service Requiring Key Stores	Description
WebLogic Server security	Enables SSL transport traffic. For more information about WebLogic Server security, see: “Secure Sockets Layer (SSL)” in “ Security Fundamentals ” in <i>Introduction to WebLogic Security</i> “ Configuring SSL ” in <i>Managing WebLogic Security</i>
Web service security	Enables message-level security for Web services. You must also ensure that the related WSSE policy files are made available to the Managed Servers. For more information about Web service security, see: “ Web Service Security ” in <i>WebLogic Workshop Online Help</i> “Overview of Web Services Security” in “ Configuring Security ” in <i>Programming WebLogic Web Services</i> <i>Building a Secure Web Service using BEA’s WebLogic Workshop</i> at: http://dev2dev.bea.com/products/wlworkshop81/articles/sws_wlw.jsp
WSRP security	Enables security for the Web Services for Remote Portlets (WSRP) feature of WebLogic Portal. By default, the WSRP security keystore is stored in the <code>wsrpKeystore.jks</code> file. For more information about WSRP security, see “ Establishing WSRP Security ” in <i>Using WSRP With WebLogic Portal</i> .
Trading Partner security	Enables the security for Trading Partners. For more information, see “ Trading Partner Integration Security ” in <i>Introducing Trading Partner Integration</i> .

Configuring the Managed Server Start Attributes

You need to configure the `serverStart` attributes for each Managed Server if you want to start the Managed Server remotely using Node Manager.

The following table defines the attributes that need to be set for each Managed Server. For information about the start attributes that can be set for a server using the WebLogic Server Administration Console, see “[ServerStart](#)” in *WebLogic Server Configuration Reference*.

Table 3-3 ServerStart Attributes for the Managed Server

Attribute	Description
Arguments	Startup arguments to use when starting this server.
BEA Home	Location of the WebLogic Platform installation.
ClassPath	Classpath to use when starting the server. For sample classpath settings for default domains, see Table 3-4 .
JavaHome	Java home directory.
Name	Name of the configuration.
Notes	Optional information that you can include to describe the configuration.
Password	Password of the username used to boot the server and perform server health monitoring.
RootDirectory	Root directory to be used to start the server.
SecurityPolicyFile	Security policy file to use when starting this server.
Username	Username to use when booting the server and performing server health monitoring.

The following table identifies the required `CLASSPATH` settings for the default domains.

Note: The classpath value may differ from that defined in the table, depending on your environment. To access the `CLASSPATH` information for your environment, start the Administration Server and capture the `CLASSPATH` information that displays to the command window (`stdout`).

Table 3-4 CLASSPATH Settings for Default Domains

Domain	CLASSPATH Value
Basic WebLogic Platform	<pre>%WL_HOME%\server\lib\weblogic_knex_patch.jar; %WL_HOME%\common\lib\log4j.jar; %WL_HOME%\server\lib\debugging.jar; %WL_HOME%\server\lib\knex.jar; %WL_HOME%\javelin\lib\javelin.jar; %WL_HOME%\server\lib\wlw-lang.jar; %JAVA_HOME%\lib\tools.jar; %WL_HOME%\server\lib\weblogic_sp.jar; %WL_HOME%\server\lib\weblogic.jar; %WL_HOME%\server\lib\ant\ant.jar; %JAVA_HOME%\jre\lib\rt.jar; %WL_HOME%\server\lib\webserviceclient.jar; %WL_HOME%\server\lib\webserviceclient+ssl.jar; %WL_HOME%\server\lib\xbean.jar; %WL_HOME%\server\lib\wlxbean.jar; %WL_HOME%\server\lib\xqrl.jar; %WL_HOME%\server\lib\netui\netui-compiler.jar; %WL_HOME%\server\lib\wli.jar; %WL_HOME%\server\lib\fop.jar; %WL_HOME%\integration\adapters\sample\lib\sample-eis.jar; %WL_HOME%\portal\lib\wps_system.jar</pre>
Basic WebLogic Server	<pre>%JAVA_HOME%\lib\tools.jar; %WL_HOME%\server\lib\weblogic_sp.jar; %WL_HOME%\server\lib\weblogic.jar; %JAVA_HOME%\jre\lib\rt.jar; %WL_HOME%\server\lib\webservices.jar</pre> <p data-bbox="381 1204 1103 1256">For more information, see “Setting the CLASSPATH” in <i>WebLogic Server Command Reference</i>.</p>

Table 3-4 CLASSPATH Settings for Default Domains (Continued)

Domain	CLASSPATH Value
Basic WebLogic Portal	<pre> %WL_HOME%\server\lib\weblogic_knex_patch.jar; %WL_HOME%\common\lib\log4j.jar; %WL_HOME%\server\lib\debugging.jar; %WL_HOME%\server\lib\knex.jar; %WL_HOME%\javelin\lib\javelin.jar; %WL_HOME%\server\lib\wlw-lang.jar; %JAVA_HOME%\lib\tools.jar; %WL_HOME%\server\lib\weblogic_sp.jar; %WL_HOME%\server\lib\weblogic.jar; %WL_HOME%\server\lib\ant\ant.jar; %JAVA_HOME%\jre\lib\rt.jar; %WL_HOME%\server\lib\webserviceclient.jar; %WL_HOME%\server\lib\webserviceclient+ssl.jar; %WL_HOME%\server\lib\xbean.jar; %WL_HOME%\server\lib\wxbean.jar; %WL_HOME%\server\lib\xqrl.jar; %WL_HOME%\server\lib\netui\netui-compiler.jar; %WL_HOME%\server\lib\wli.jar; %WL_HOME%\server\lib\fop.jar; %WL_HOME%\integration\adapters\sample\lib\sample-eis.jar; %WL_HOME%\portal\lib\wps_system.jar </pre>

Table 3-4 CLASSPATH Settings for Default Domains (Continued)

Domain	CLASSPATH Value
Basic WebLogic Integration	<pre>%WL_HOME%\server\lib\weblogic_knex_patch.jar; %WL_HOME%\common\lib\log4j.jar; %WL_HOME%\server\lib\debugging.jar; %WL_HOME%\server\lib\knex.jar; %WL_HOME%\javelin\lib\javelin.jar; %WL_HOME%\server\lib\wlw-lang.jar; %JAVA_HOME%\lib\tools.jar; %WL_HOME%\server\lib\weblogic_sp.jar; %WL_HOME%\server\lib\weblogic.jar; %WL_HOME%\server\lib\ant\ant.jar; %JAVA_HOME%\jre\lib\rt.jar; %WL_HOME%\server\lib\webserviceclient.jar; %WL_HOME%\server\lib\webserviceclient+ssl.jar; %WL_HOME%\server\lib\xbean.jar; %WL_HOME%\server\lib\wlxbean.jar; %WL_HOME%\server\lib\xqrl.jar; %WL_HOME%\server\lib\netui\netui-compiler.jar; %WL_HOME%\server\lib\wli.jar; %WL_HOME%\server\lib\fop.jar; %WL_HOME%\integration\adapters\sample\lib\sample-eis.jar</pre>
Basic WebLogic Workshop	<pre>%WL_HOME%\server\lib\weblogic_knex_patch.jar; %WL_HOME%\common\lib\log4j.jar; %WL_HOME%\server\lib\debugging.jar; %WL_HOME%\server\lib\knex.jar; %WL_HOME%\javelin\lib\javelin.jar; %WL_HOME%\server\lib\wlw-lang.jar; %JAVA_HOME%\lib\tools.jar; %WL_HOME%\server\lib\weblogic_sp.jar; %WL_HOME%\server\lib\weblogic.jar; %WL_HOME%\server\lib\ant\ant.jar; %JAVA_HOME%\jre\lib\rt.jar; %WL_HOME%\server\lib\webserviceclient.jar; %WL_HOME%\server\lib\webserviceclient+ssl.jar; %WL_HOME%\server\lib\xbean.jar; %WL_HOME%\server\lib\wlxbean.jar; %WL_HOME%\server\lib\xqrl.jar; %WL_HOME%\server\lib\netui\netui-compiler.jar; %WL_HOME%\server\lib\wli.jar; %WL_HOME%\server\lib\fop.jar; %WL_HOME%\integration\adapters\sample\lib\sample-eis.jar</pre>

Configuring Node Manager

Node Manager enables you to perform common operations for a Managed Server, regardless of its location with respect to its Administration Server. You can use Node Manager to perform the following operations:

- Start and stop remote Managed Servers.
- Monitor the health of Managed Servers and automatically kill server instances whose health state is “failed.”
- Automatically restart Managed Servers that have failed, or have shutdown unexpectedly due to a system crash or reboot.

To take advantage of Node Manager capabilities, you must run a Node Manager process on each machine that hosts Managed Servers, and configure Node Manager to:

- Accept commands from a remote host

For example, you must configure a machine to use Node Manager and specify a value other than localhost for the listen address, as described in [“Considerations for Configuring and Targeting Resources”](#) on page 3-3.

- Support production security components
- Tailor the shutdown and monitoring behavior

For complete details about each of the steps required to configure Node Manager in a production environment, see “Configuration Checklist (Production Environment)” in “Configuring Node Manager” in [“Configuring, Stopping, and Starting Node Manager”](#) in *Configuring and Managing WebLogic Server*.

Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline

The following example illustrates how to create a WebLogic Platform domain that contains a single cluster, as illustrated in [“Single-Cluster Platform Domain Example”](#) on page 2-9, using WLST Offline.

Steps include:

- [“Define WLST Variables”](#) on page 3-22
- [“Open the Template for Domain Creation”](#) on page 3-22

- “Configure an Administrative Username and Password” on page 3-23
- “Configure the Administration Server” on page 3-23
- “Configure the Asynchronous Dispatcher Queues Required by the Application” on page 3-23
- “Configure the JDBC Connection Pools” on page 3-24
- “Configure the Cluster, Managed Servers, and Machines” on page 3-25
- “Set Domain Options” on page 3-27
- “Write the Domain and Close the Template” on page 3-27

For more information about WLST Offline, see the documentation and examples provided with the WLST Offline kit at the following URL:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

Define WLST Variables

You can define variables in the WLST Offline script and reference them using the WLST Offline commands to facilitate reuse of the scripts in different target environments. For example, you can “hard-code” the variable values within the script or pass the values to the WLST Offline script from the command line.

For example, if you define the following variables within the WLST Offline script to specify the WebLogic home directory, domain name, and domain template name:

```
weblogic_home="/bea/weblogic81"  
domain_template="platform.jar"
```

You can then reference the variables in the script instead of the values. For example:

```
readTemplate(weblogic_home + '/common/templates/domains'+ domain_template)
```

Note: For readability, the following code excerpts use actual values and not variables.

Open the Template for Domain Creation

The following code excerpt opens the Basic WebLogic Platform Domain template for domain creation and sets the name of the domain using the `cmo` variable. The `cmo` variable stores the Current Management Object, which is set to the configuration bean instance to which you navigate using WLST Offline—in this case the root of all configuration management objects, `DomainMBean`.

```
readTemplate('/bea/weblogic81/common/templates/domains/platform.jar')
```

Configure an Administrative Username and Password

The following code excerpt sets the password for the pre-configured administrative account.

```
cd('/Security/platform/User/weblogic')
cmo.setPassword('password')
```

Configure the Administration Server

The following code excerpt configures the Administration Server by setting the listen address and port, enabling SSL, and setting the SSL port. To review considerations for configuring the Administration Server, see [“Considerations for Configuring and Targeting Resources” on page 3-3](#).

```
cd('/Server/cgServer')
cmo.setListenAddress('host')
cmo.setListenPort(9301)

cd('/SSL/cgServer')
cmo.setEnabled(1)
cmo.setListenPort(9302)
```

Configure the Asynchronous Dispatcher Queues Required by the Application

The following code excerpt configures a set of asynchronous dispatcher queues and their associated error queues, as required by the application. To determine if you need to create asynchronous dispatcher queues, review the contents of the `wlw-manifest.xml` file, which provides information about the server resources referenced by the application, as described in [“Adding Application Resources Required by the WebLogic Workshop Runtime” on page 3-13](#).

Tip By creating the asynchronous queues *before* you create the cluster, the queue will be configured automatically as a distributed queue with the same JNDI name.

```
queueNames = 'IntApp.queue.AsyncDispatcher', \
  'PortApp.queue.AsyncDispatcher'

# Predefined JMS Server included in the domain template
cd('/JMSServer/cgJMSServer')

for qName in queueNames:
    errqName = qName + "_error"
```

Creating and Configuring the WebLogic Domain

Create asynchronous dispatcher error queue

```
create(errqName, 'JMSQueue')
cd ('JMSQueue/' + errqName)
cmo.setJNDIName(errqName)
errqMBean = cmo
cd ('../..')
```

Create asynchronous dispatcher queue

```
create(qName, 'JMSQueue')
cd ('JMSQueue/' + qName)
cmo.setJNDIName(qName)
```

Associate asynchronous dispatcher error queue

```
cmo.setErrorDestination(errqMBean)
cd ('../..')
```

Configure the JDBC Connection Pools

The following code excerpt configures the database type and related information to specify the **production** database for each of the four default JDBC connection pools, including bpmArchPool, cgJMSPool-nonXA, cgPool, and portalPool. To review considerations for configuring JDBC connection pools, see [“Considerations for Configuring and Targeting Resources” on page 3-3](#).

```
cd ('/JDBCConnectionPool')
nonXApools = 'cgJMSPool-nonXA',
for pool in nonXApools:
    cd (pool)
    cmo.setDriverName('oracle.jdbc.driver.OracleDriver')
    cmo.setUsername('username')
    cmo.setPassword('password')
    cmo.setDbmsHost('myDBHost')
    cmo.setDbmsPort('1521')
    cmo.setDbmsName('myDB')
    cd ('..')

XApools = 'cgPool', 'bpmArchPool', 'portalPool'
for pool in XApools:
    cd (pool)
    cmo.setDriverName('oracle.jdbc.xa.client.OracleXADatasource')
    cmo.setUsername('user')
    cmo.setPassword('password')
    cmo.setDbmsHost('myDBHost')
```



```
cmo.setDbmsPort('5321')
cmo.setDbmsName('myDB')
cmo.setSupportsLocalTransaction(1)
cmo.setKeepXAConnTillTxComplete(1)
cd ('..')
```

Configure the Cluster, Managed Servers, and Machines

The following code excerpt performs the following tasks:

- Configures one cluster.
- Configures two machines.
- Configures the Node Manager name, listen address, start properties—including `BEA_HOME`, `JAVA_HOME`, the directory root, and the `CLASSPATH` values—and the file containing the security policy for starting the server. For more information about configuring Node Manager, see [“Configuring Node Manager” on page 3-21](#).
- Configures four Managed Servers, targeting them to a machine, and enables SSL to enable secure access to application resources.
- Tunes the default execution threads. For more information, see [“Tuning the Default Execute Queue Threads”](#) in *WebLogic Server Performance and Tuning*.

```
cd ('/')
# Define the servers
clusterConfig = 'platformCluster','multicast_addr',9101
ms1 = 'ms1','host1',9111,9112,'machine1'
ms2 = 'ms2','host2',9121,9122,'machine2'
ms3 = 'ms3','host3',9131,9132,'machine1'
ms4 = 'ms4','host4',9141,9142,'machine2'
mss = ms1, ms2, ms3, ms4

#Create the cluster
cl_name, mc_addr, mc_port = clusterConfig
cluster = create(cl_name, 'Cluster')

#Configure the machines
machine = None
prev_ms_machine = ""
cl_addr = ""
for msConfig in mss:
    ms_name, ms_addr, ms_port, ms_ssl_port, ms_machine = msConfig
    if ms_machine != "" and ms_machine != prev_ms_machine:
        prev_ms_machine = ms_machine
        machine = create(ms_machine, 'Machine')
```

Creating and Configuring the WebLogic Domain

```
#Configure the Node Manager name and listen address
cd('Machine/' + ms_machine + '/NodeManager/' + ms_machine)
cmo.setName(ms_machine)
cmo.setListenAddress(ms_addr)
cd('../ ../../..')

#Configure the Managed Server, target it to a machine
ms = create(ms_name, 'Server')

cd('Server/' + ms_name)
cmo.setListenAddress(ms_addr)
cmo.setListenPort(ms_port)
if machine != None:
    cmo.setMachine(machine)
cd('SSL/' + ms_name)
cmo.setEnabled(1)
cmo.setListenPort(ms_ssl_port)
cd('../ ../../..')
assign('Server', ms_name, 'Cluster', cl_name)

# Configure Node Manager StartServer properties
create(ms_name, 'ServerStart')
cd('ServerStart/' + ms_name)
cmo.setBeaHome('/bea')
cmo.setJavaHome('/bea/weblogic81/jrocket81sp4_142_05')
cmo.setRootDirectory('/bea/user_projects/domains/platform')
# See Table 3-4 for CLASSPATH settings
cmo.setClassPath('classpath')
cmo.setSecurityPolicyFile('/bea/weblogic81/server/lib/weblogic.policy')
cmo.setArguments('-Xms512m -Xmx512m -da -Dwlw.iterativeDev=false
-Dwlw.testConsole=false -Dwlw.logErrorsToConsole=true
-Dweblogic.ProductionModeEnabled=true')
cmo.setUsername('user')
cmo.setPassword('password')

cd('../ ../../..')
# Tune execution threads
cd('ExecuteQueue/weblogic.kernel.Default')
cmo.setThreadCount(15)
cmo.setThreadsIncrease(5)

cd('../ ../../..')
# Accumulate addresses and ports for setting cluster address
cl_addr = cl_addr + ms_addr + ':' + str(ms_port) + ','

# Set the cluster address, multicast address, and multicast port
cl_addr = cl_addr[:-1] # slice off trailing ','
cluster.setClusterAddress(cl_addr)
```

```
cluster.setMulticastAddress(mc_addr)
cluster.setMulticastPort(mc_port)
```

Configure the Web Proxy Server

The following code excerpt configures the Web proxy server. For more information, see [“Using Load Balancers and Web Proxy Servers” on page 6-1](#).

```
create('platproxy', 'Server')
cd('/Server/platproxy')
cmo.setListenAddress('host')
cmo.setListenPort(9431)
# Set this server as the proxy server for the cluster
cd('/Cluster/platformcluster')
cmo.setProxyServer('platproxy')
```

Set Domain Options

The following code excerpt sets the following domain options:

- `CreateStartMenu`—Specifies whether to create a Start Menu shortcut on a Windows platform.
- `JavaHome`—Specifies the home directory for the SDK to be used when starting the server.
- `OverwriteDomain`—Specifies whether to allow an existing domain to be overwritten. The default is `false`.
- `ServerStartMode`—Specifies the mode to use when starting the server for the newly created domain. This value can be `dev` (development) or `prod` (production). This value should be set to `prod`.

```
setOption('OverwriteDomain', 'true')
setOption('CreateStartMenu', 'false')
setOption('ServerStartMode', 'prod')
setOption('JavaHome', '/bea/weblogic81/jrockit81sp4_142_05')
```

Write the Domain and Close the Template

The following code excerpt writes the domain to the specified directory and closes the configuration template.

```
writeDomain('/bea/user_projects/domains/platform')
closeTemplate()
```

Example: How to Configure a Multi-Domain Environment Using WLST Offline

This section describes how to configure an environment that consists of two domains—a Basic WebLogic Integration Domain and a Basic WebLogic Portal Domain—as illustrated in “[Multi-Domain Example](#)” on page 2-14.

Steps include:

- “[Create the Basic WebLogic Integration Domain](#)” on page 3-28
- “[Create the Basic WebLogic Portal Domain](#)” on page 3-28

For more information about using WLST Offline, see the documentation and examples provided with the WLST Offline kit at the following URL:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

Create the Basic WebLogic Integration Domain

The process used to configure the Basic WebLogic Integration domain using WLST Offline is similar to the process illustrated in “[Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline](#)” on page 3-21, with the following exceptions:

- When opening the template to create the domain, as described in “[Open the Template for Domain Creation](#)” on page 3-22, specify the Basic WebLogic Integration Domain template. For example:

```
readTemplate('/bea/weblogic81/common/templates/domains/wli.jar')
```

- When configuring the asynchronous dispatcher queues required by the application, as described in “[Configure the Asynchronous Dispatcher Queues Required by the Application](#)” on page 3-23, define the `queueNames` list as follows:

```
queueNames = 'IntApp.queue.AsyncDispatcher',
```

- When configuring the JDBC connection pools, as described in “[Configure the JDBC Connection Pools](#)” on page 3-24, define the XA connection pools list as follows:

```
XApools = 'cgPool', 'bpmArchPool'
```

Create the Basic WebLogic Portal Domain

The process used to configure the Basic WebLogic Portal domain using WLST Offline is similar to the process illustrated in “[Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline](#)” on page 3-21, with the following exceptions:

Example: How to Configure a Multi-Domain Environment Using the Configuration Wizard in Silent Mode

- When opening the template to create the domain, as described in “[Open the Template for Domain Creation](#)” on page 3-22, specify the Basic WebLogic Portal Domain template. For example:

```
readTemplate('/bea/weblogic81/common/templates/domains/wlp.jar')
```

- When configuring the asynchronous dispatcher queues required by the application, as described in “[Configure the Asynchronous Dispatcher Queues Required by the Application](#)” on page 3-23, define the `queueNames` list as follows:

```
queueNames = 'PortApp.queue.AsyncDispatcher',
```

- When configuring the JDBC connection pools, as described in “[Configure the JDBC Connection Pools](#)” on page 3-24, define the `XAPools` list as follows:

```
XAPools = 'cgPool', 'portalPool'
```

Example: How to Configure a Multi-Domain Environment Using the Configuration Wizard in Silent Mode

Clusterizing End2End on WebLogic Platform 8.1 on the dev2dev Web site provides an example of configuring a target domain using the Configuration Wizard in silent mode. You can access this article, and download its associated samples files at:

http://dev2dev.bea.com/products/wlpplatform81/articles/clusterizing_E2E.jsp

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

This article describes how to promote and extend the WebLogic Platform Tour from a development to a production environment, and describes how to configure the following two domains using the Configuration Wizard silent mode:

- WebLogic Portal domain for hosting WebLogic Portal applications
- WebLogic Integration domain for hosting WebLogic Integration applications

Note: This environment is depicted in “[Multi-Domain Example](#)” on page 2-14.

To view the Configuration Wizard silent mode scripts, download the `E2ECluster.zip` file that contains the sample code and extract the following files:

- `integration_cluster_domain.cws`—Creates a clustered domain for the sample using the Basic WebLogic Integration Domain template.
- `portal_cluster_domain.cws`—Creates a clustered domain for the sample using the Basic WebLogic Portal Domain template.

Note: The scripts are designed to be called from an Ant script. They reference *properties*, such as @DOMAIN_TEMPLATE@, that are resolved in a properties file imported to the Ant script. Using Ant scripting and properties in this way facilitates automation and reuse of the scripts in different target environments.

Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline

When you want to configure more than one cluster in a WebLogic Platform domain, you must explicitly retarget some of the resources, services, and applications. You must do this because of the following:

- The Configuration Wizard and WLST Offline support autoconfiguration for a single-cluster domain only. Autoconfiguration for a second cluster is not supported.
- There can be only one WebLogic Integration cluster in a domain. That is, there can be no WebLogic Integration resources in a second cluster.

Tip A strategy for configuring two clusters in a WebLogic Platform domain is to leverage autoconfiguration for a single cluster as much as possible. You can designate the first cluster in the domain to be the WebLogic Integration cluster and the second cluster to be the WebLogic Portal cluster. Because a WebLogic Integration cluster requires JMS queues and topics, this strategy leverages autoconfiguration to set up these resources. You then target the remaining resources, services, and applications correctly. And, finally, you edit the `config.xml` file to add JMS resources for the second, WebLogic Portal cluster.

The following example illustrates how to create a WebLogic Platform domain that contains two clusters, as illustrated in [“Multi-Cluster Platform Domain Example” on page 2-17](#), using WLST Offline.

Steps include:

- [“Create a Domain With Two Clusters” on page 3-31](#)
- [“Correct Targets for the Resources, Services, and Applications” on page 3-33](#)
- [“Manually Edit the config.xml File to Add JMS Resources for the Second Cluster” on page 3-35](#)

Note: For readability, the following code excerpts use actual values and not variables. For information about using WLST variables, see [“Define WLST Variables” on page 3-22](#).

For more information about using WLST Offline, see the documentation and examples provided with the WLST Offline kit at the following URL:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

Create a Domain With Two Clusters

The process used to configure a multi-cluster domain using WLST Offline is similar to the process illustrated in “[Example: How to Configure a Single-Cluster Platform Domain Using WLST Offline](#)” on page 3-21, except that you need to create two clusters rather than a single cluster.

For example, the following code excerpt performs the following tasks:

- Configures two clusters—WebLogic Integration cluster and a WebLogic Portal cluster.
- Configures four machines, two for the WebLogic Integration cluster, two for the WebLogic Portal cluster.
- Configures the Node Manager name, listen address, start properties—including `BEA_HOME`, `JAVA_HOME`, the directory root, and the `CLASSPATH` values—and the file containing the security policy for starting the server. For more information about configuring Node Manager, see “[Configuring Node Manager](#)” on page 3-21.
- Configures the Managed Servers, targeting them to a machine, and enables SSL to enable secure access to application resources.
- Configures the JMS JDBC Store for the WebLogic Portal cluster.

Compare this code excerpt to the equivalent excerpt for the single cluster domain, described in “[Configure the Cluster, Managed Servers, and Machines](#)” on page 3-25.

```
cd ('/')
# Define the servers
wli_ms1 = 'wlms1', 'host1', 9111, 9112, 'wlimachine1'
wli_ms2 = 'wlms2', 'host2', 9121, 9122, 'wlimachine2'
wli_mss = wli_ms1, wli_ms2
wli_clusterConfig = 'wlicluster', 'multicast1_addr', 9101, wli_mss

wlp_ms1 = 'wlpms1', 'host3', 9211, 9212, 'wlpmachine1'
wlp_ms2 = 'wlpms2', 'host4', 9221, 9222, 'wlpmachine2'
wlp_mss = wlp_ms1, wlp_ms2
wlp_clusterConfig = 'wlpcluster', 'multicast2_addr', 9201, wlp_mss

#Create the two clusters
clusterConfigs = wli_clusterConfig, wlp_clusterConfig

machines_created = ""
```

Creating and Configuring the WebLogic Domain

```
for clusterConfig in clusterConfigs:
    cl_name, mc_addr, mc_port, mss = clusterConfig
    cluster = create(cl_name, 'Cluster')

#Configure the machines
machine = None
cl_addr = ""

for msConfig in mss:
    ms_name, ms_addr, ms_port, ms_ssl_port, ms_machine = msConfig

    # check for existence of ms_machine
    create_this_machine = 'yes'
    if machines_created == "":
        machines_created = ms_machine
    else:
        if machines_created.find(ms_machine) == -1:
            machines_created = machines_created + "," + ms_machine
        else:
            create_this_machine = 'no'

    if ms_machine != "" and create_this_machine == 'yes':
        machine = create(ms_machine, 'Machine')

    #Configure the Node Manager name and listen address
    cd('Machine/' + ms_machine + '/NodeManager/' + ms_machine)
    cmo.setName(ms_machine)
    cmo.setListenAddress(ms_addr)
    cd('../ ../../..')

#Configure the Managed Server, target it to a machine
ms = create(ms_name, 'Server')
ms.setListenAddress(ms_addr)
ms.setListenPort(ms_port)

if machine != None:
    ms.setMachine(machine)

cd('Server/' + ms_name + '/SSL/' + ms_name)
cmo.setEnabled(1)
cmo.setListenPort(ms_ssl_port)
cd('../ ../../..')

assign('Server', ms_name, 'Cluster', cl_name)

# Configure Node Manager StartServer properties
cd('Server/' + ms_name)
create(ms_name, 'ServerStart')
cd('ServerStart/' + ms_name)
```


Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline

```
cmo.setBeaHome('/bea')
cmo.setJavaHome('/bea/weblogic81/jrocket81sp4_142_05')
cmo.setRootDirectory('/bea/user_projects/domains/platform')
# See Table 3-4 for CLASSPATH settings
cmo.setClassPath('classpath')
cmo.setSecurityPolicyFile('/bea/weblogic81/server/lib/weblogic.policy')
cmo.setArguments('-Xms512m -Xmx512m -da -Dwlw.iterativeDev=false
-Dwlw.testConsole=false -Dwlw.logErrorsToConsole=true
-Dweblogic.ProductionModeEnabled=true')
cmo.setUsername('user')
cmo.setPassword('password')
cd('../ ../../..')
# Accumulate addresses and ports for setting cluster address
cl_addr = cl_addr + ms_addr + ':' + str(ms_port) + ','

# Set the cluster address, multicast address, and multicast port
cl_addr = cl_addr[:-1] # slice off trailing ','
cluster.setClusterAddress(cl_addr)
cluster.setMulticastAddress(mc_addr)
cluster.setMulticastPort(mc_port)

#Configure the JMS JDBC Store for the WebLogic Portal Cluster
cd('/JDBCConnectionPool/cgJMSPool-nonXA')
connPoolMBean = cmo
cd('/JMSTemplate/TemporaryTplmt')
jmsTemplateMBean = cmo
cd('/')
jdbcstorePrefix = "cgJMSStore_wlp_auto_"
a = ['1', '2']
for n in a:
    storename = jdbcstorePrefix + n
    store = create(storename, "JMSJDBCStore")
    store.setConnectionPool(connPoolMBean)
    store.setPrefixName('wlp_' + n)
```

Correct Targets for the Resources, Services, and Applications

The Configuration Wizard and WLST Offline support autoconfiguration for single-cluster domains only. When creating two clusters, you need to manually correct the targets assigned for the resources, services, and applications, as described in the following procedure. (For more information about autoconfiguration, see [“Autoconfiguration Using the Configuration Wizard and WLST Offline”](#) on page 3-12.

Note: For default targeting requirements, refer to [“Deployment Targeting Reference - Single Cluster WebLogic Platform Domain”](#) on page A-4. To identify resources, services, and application product components, see [“Default Domain Resource Reference - By Product Component”](#) on page A-14.

1. Correct the targets for the system resources.

For example, unassign the `portalPool` and `bpmArchPool` JDBC connection pools from the WebLogic Integration and WebLogic Portal clusters, respectively.

```
unassign('JDBCConnectionPool', 'portalPool', 'Target', wliClusterName)
unassign('JDBCConnectionPool', 'bpmArchPool', 'Target', wlpClusterName)
```

2. Correct the targets for the WebLogic services.

For example, unassign the WebLogic Integration startup and shutdown classes from the WebLogic Portal cluster.

```
WLIStartupShutdownClasses = (('ShutdownClass','WLI Shutdown Class'),\
    ('StartupClass','WLI Startup Class'),\
    ('StartupClass','WLI Post-Activation Startup Class'))
for classNVpair in WLIStartupShutdownClasses:
    cname,cvalue=classNVpair
    unassign(cname, cvalue, 'Target', wlpClusterName)
```

3. Correct the targets for the applications.

- a. Unassign all applications from the following resources: Administration Server, First Managed Server defined for the WebLogic Integration cluster, and WebLogic Integration and WebLogic Portal clusters

For example:

```
unassignAll('Application', 'Target', adminName)
unassignAll('Application', 'Target', wlims1Name)
unassignAll('Application', 'Target', wliClusterName)
unassignAll('Application', 'Target', wlpClusterName)
```

- b. Re-assign the applications to the appropriate targets.

WLST Offline does not support resource names that contain period (“.”) or slash (“/”) characters. Therefore, you must temporarily substitute all period (“.”) and slash (“/”) characters in application names when assigning them to targets. Once assigned, you must restore the original application name for the application to function properly within the domain. For example, in the following excerpt, the following temporary substitutions have been made:

- `.workshop/worklist/EJB/ProjectBeans` has been temporarily renamed as follows:
`_workshop_worklist_EJB_ProjectBeans`
- `.workshop/worklist/EJB/GenericStateless` has been temporarily renamed as follows:
`_workshop_worklist_EJB_GenericStateless`

For example:

```
assign('Application', 'B2BDefaultWebApp', 'Target', adminName)
assign('Application', 'WLI AI RAR Upload', 'Target', adminName)
assign('Application', 'wliconsole', 'Target', adminName)
assign('Application', 'B2BDefaultWebApp', 'Target', wliClusterName)
assign('Application', '_workshop_worklist_EJB_ProjectBeans', \
'Target', wliClusterName)
assign('Application', '_workshop_worklist_EJB_GenericStateless', \
'Target', wliClusterName)
assign('Application', 'worklist', 'Target', wliClusterName)
assign('Application', 'WLIAdmin', 'Target', wliClusterName)
assign('Application', 'WLI Admin Helper', 'Target', wliClusterName)
...
```

Manually Edit the config.xml File to Add JMS Resources for the Second Cluster

Manually edit the `config.xml` file to create the following resources for the second cluster, in this example, the WebLogic Portal cluster:

- JMS Server for each Managed Server

For example:

```
<JMSServer Name="cgJMSServer_wlp_auto_1"
  Store="cgJMSStore_wlp_auto_1" Targets="wlpms1 (migratable)">
  <JMSQueue JNDIName="jws.queue_wlp_auto_1"
    Name="cgJWSQueue_wlp_auto_1" RedeliveryLimit="2"
    StoreEnabled="default"/>
</JMSServer>
<JMSServer Name="cgJMSServer_wlp_auto_2"
  Store="cgJMSStore_wlp_auto_2" Targets="wlpms2 (migratable)">
  <JMSQueue JNDIName="jws.queue_wlp_auto_2"
    Name="cgJWSQueue_wlp_auto_2" RedeliveryLimit="2"
    StoreEnabled="default"/>
</JMSServer>
```

- Distributed queues

For example:

```
<JMSDistributedQueue JNDIName="jws.queue"
  Name="dist_cgJWSQueue_wlp_auto" Targets="wlpcluster">
  <JMSDistributedQueueMember JMSQueue="cgJWSQueue_wlp_auto_1"
    Name="cgJWSQueue_wlp_auto_1_OF_cgJMSServer_wlp_auto_1"/>
  <JMSDistributedQueueMember JMSQueue="cgJWSQueue_wlp_auto_2"
    Name="cgJWSQueue_auto_2_OF_cgJMSServer_wlp_auto_2"/>
</JMSDistributedQueue>
```

Creating and Configuring the WebLogic Domain

Configuring the Production Database

When deploying to a production environment, you need to set up an enterprise-level database instance and define the database tables.

This section describes how to configure the WebLogic Portal and WebLogic Integration database tables, and promote the existing data to the target environment.

Note: This section describes configuring databases to meet the requirements of the WebLogic Platform product runtime. Configuring databases for application-specific requirements is beyond the scope of this document. It is assumed that users will ensure that databases are properly set up to support application business logic and application data requirements.

Topics include:

- [Creating and Preparing a Production Database](#)
- [Creating Conversational State Database Tables](#)
- [Promoting Database Information to the Production Database](#)
- [Example: How to Load the Domain Database Using WLST Offline](#)
- [Example: How to Load the Application Database Using Ant](#)
- [Example: How to Create the Conversational State Database Tables Using Ant](#)

Creating and Preparing a Production Database

To create the database schema and load a database for use with WebLogic Platform applications, perform the following steps:

1. Review [Supported Configurations for WebLogic Platform 8.1](#) to ensure that your configuration is supported.
2. Create your vendor database(s).

If you want to use the WebLogic Portal behavior event tracking in a production environment, consider using a separate database for behavior event tracking. For more information, see [WebLogic Portal Database Administration Guide](#).

3. Prepare the database as required by the WebLogic Platform applications.

BEA provides several sample initialization scripts that need to be modified and run on the vendor database before using the database with WebLogic Platform components. The initialization scripts load the appropriate information into the database tables.

To prepare the database for WebLogic Platform applications that implement WebLogic Portal features, see the following sections in the [WebLogic Portal Database Administration Guide](#), based on your vendor type:

- [“Using a Microsoft SQL Server Database”](#)
- [“Using an Oracle Database”](#)
- [“Using a Sybase Database”](#)
- [“Using a DB2 Database”](#)

To prepare the database for WebLogic Platform applications that implement WebLogic Integration features, see [“Configuring a Production Database”](#) in [Managing WebLogic Integration Solutions](#).

Creating Conversational State Database Tables

If your application uses Web services or Timer controls, WebLogic Workshop uses database tables in the development domain to store the conversational state.

WebLogic Workshop lists the conversational state tables in the `/META-INF/wlw-manifest.xml` file with the following element name `<con:conversation-state-table>`.

For each conversational state table element defined in the `wlw-manifest.xml` file, you must configure a corresponding table in the target database. SQL scripts for defining the conversational state tables for specific databases are described in Step 2 of “To Manually Create Required Resources on the Production Server” in [“How Do I: Deploy a WebLogic Workshop Application to a Production Server?”](#) in the [WebLogic Workshop Online Help](#).

For example, for Oracle, you can use the following SQL script to create the conversational state tables, where `<TableName>` specifies the name of the table that appears in the `<con:conversation-state-table>` element.

```
CREATE TABLE <TableName>(
  CG_ID varchar(768) not null,
  LAST_ACCESS_TIME number(19,0),
  CG_DATA BLOB,
  PRIMARY KEY (CG_ID)
)
```

For an example of creating the conversational state database tables using Ant, see [“Example: How to Create the Conversational State Database Tables Using Ant”](#) on page 4-9

Promoting Database Information to the Production Database

The following sections describe the steps required to promote existing database information to the enterprise-level production database.

- [“Promoting WebLogic Portal Datasync Information”](#) on page 4-3
- [“Promoting LDAP and Portal Database Data”](#) on page 4-5
- [“Promoting WebLogic Integration Application Database Information”](#) on page 4-5

Promoting WebLogic Portal Datasync Information

WebLogic Portal datasync information consists of user segments, content selectors, campaigns, discounts, and the following property sets: user profile, catalog, event, and session and request. This information must be managed carefully when moving from development to production, and bootstrapped as a separate process during deployment, as described in [“Steps to Deploy the Application”](#) on page 8-7.

In a development environment, portal application datasync information is created and stored as a special Datasync Project within WebLogic Workshop. The files are stored in the `META-INF/data` directory of the portal application and exposed in WebLogic Workshop in the `<portalApplication>/data` directory. Datasync data is modified by developers, as well as business users and portal administrators, who can modify user segments, campaigns, placeholders, and content selectors within the WebLogic Administration Portal, and is updated by both WebLogic Workshop and WebLogic Administration Portal.

In a production environment, users need to be able to modify portal definitions using the WebLogic Administration Portal. The method that you use to enable updates to the datasync information in the production environment depends, in part, on how the portal application is deployed, as described in the following table.

Table 4-1 How Datasync Information is Updated Based on Archive Type

If the portal application is deployed as...	Then...
A compressed EAR file	All datasync data must be loaded from the file system into the database so the application can be updated. When an application is deployed, if the JDBC datasync repository is empty, then the files in the archive will be used to bootstrap, or initialize, the database. Datasync information is stored in the following tables: DATA_SYNC_APPLICATION, DATA_SYNC_ITEM, DATA_SYNC_SCHEMA_URI, and DATA_SYNC_VERSION.
An exploded EAR	The deployed portal application on the Administration Server is the primary store of datasync definitions. You can update the information from any server in the domain using the WebLogic Administration Portal, unless the EAR directory is read-only. If you update the datasync information using a Managed Server, the changes are automatically synchronized with the primary store.

When you initially deploy a WebLogic Portal application to a production environment, as a compressed or exploded archive, the datasync information is bootstrapped, or initialized in the database automatically.

In addition, each portal application contains a Datasync Web application that enables you to view the datasync repository and upload new content. You can access the Datasync Web application using the following URL: `http://servername:port/appNameDataSync`, where *servername* and *port* are the name and port number for the server and *appName* is the WebLogic Portal application name.

If you need to make incremental updates to the portal data, you can use the Datasync Web application to load new information as part of a redeploy operation or independently, using a separate standalone JAR file. Extreme care must be used so as not to remove or inadvertently modify valid data. You can also retrieve datasync information from the production environment and save it to your staging environment.

For more information about promoting and retrieving datasync information, see [“Using the Datasync Web Application”](#) in the WebLogic Portal *Production Operations User Guide*.

Promoting LDAP and Portal Database Data

The WebLogic Portal LDAP data includes global roles, enterprise application roles, Web application roles (such as delegated administration and visitor entitlement roles), delegated administration assignments/definitions, and visitor entitlement assignments/definitions. The database data includes any data that is created or modified with the WebLogic Administration Portal.

The WebLogic Portal Propagation Utility enables you to promote application LDAP and database data to the production environment. You can also retrieve LDAP and portal database information from the production environment and save it to your staging environment.

The Propagation Utility is a portal Web application packaged inside an enterprise application archive EAR. The utility must be deployed on both the source domain and the target domain. Contact BEA Support to obtain the latest copy of the Propagation Utility.

In a clustered environment, you should only promote to the Administration Server, which will automatically promote data to the Managed Servers. When propagating database data:

- The source and destination database types must be the same and you must be able to connect to both simultaneously.
- You must have a non-XA database driver for the database installed on the source server.

For more information about using the Propagation Utility to promote LDAP and portal database data, see [“Using the Propagation Utility”](#) in the WebLogic Portal *Production Operations User Guide*.

Promoting WebLogic Integration Application Database Information

In addition to the WebLogic Integration database tables, you might also need to create or promote additional database information based on your application requirements, as defined in the following table.

Table 4-2 Additional WebLogic Integration Production Database Requirements

If you your application uses...	You must...
Web service or Timer controls	Create the database tables that store conversational state information, as described in “Creating Conversational State Database Tables” on page 4-2 .
Trading partner management data	<p>If the server is not running, use the Bulk Loader to import the information, as described in “Using the Trading Partner Bulk Loader” in <i>Managing WebLogic Integration Solutions</i>.</p> <p>If the server is running, use the WebLogic Integration Administration Console to ensure that the running servers have consistent data in their memory cache, as described in “Trading Partner Management” in <i>Managing WebLogic Integration Solutions</i>.</p>

Example: How to Load the Domain Database Using WLST Offline

The following example illustrates how to load the domain database using WLST Offline. Steps include:

- [“Open the Template” on page 4-6](#)
- [“Configure the JDBC Connection Pool” on page 4-7](#)
- [“Load the Database” on page 4-7](#)
- [“Close the Template” on page 4-7](#)

For more information about WLST Offline, see the documentation and examples provided with the WLST Offline kit at the following URL:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

Note: For readability, the following code excerpts use actual values and not variables. For information about using WLST variables, see [“Define WLST Variables” on page 3-22](#).

Open the Template

The following code excerpt opens the domain template that you want to use to load the database.

```
readTemplate ('/bea/weblogic81/common/templates/domains/platform.jar')
```

Configure the JDBC Connection Pool

The following code excerpt configures a predefined JDBC connection pool, `cgJMSPool-nonXA` using an Oracle database.

```
existingPoolName = 'cgJMSPool-nonXA'
cd ('/JDBCConnectionPool/' + existingPoolName)
cmo.setDriverName('oracle.jdbc.driver.OracleDriver')
cmo.setUserName('user')
cmo.setPassword('password')
cmo.setDbmsHost('myDBhost')
cmo.setDbmsPort('1521')
cmo.setDbmsName('myDB')
```

Load the Database

The following code excerpt loads the database for the specified JDBC connection pool.

```
loadDB('9i', existingPoolName)
```

Close the Template

The following code excerpt closes the template without creating the domain.

```
closeTemplate()
```

Example: How to Load the Application Database Using Ant

The following example illustrates how to load the application database using Ant. For more information about Ant, see:

- “Using Ant Tasks to Configure a WebLogic Server Domain” in the *WebLogic Server Command Reference*.
- The Apache Ant Project Web site at: <http://ant.apache.org>

Define the Properties in the Ant Script

To facilitate automation and reuse of the scripts in different target environments, the code excerpt references *properties*, such as `${scripts.dir}`, that are resolved in a properties file imported to the Ant script. For example, the following properties are defined in a separate properties file, `myprops.properties`:

```
scripts.dir=scripts
appldbaccnt.type=9i
```

Configuring the Production Database

```
db.host=host
db.port=1531
db.dbname=ora920
db.driver=oracle.jdbc.driver.OracleDriver
db.sqltask.url=jdbc:oracle:thin:@${db.host}:${db.port}:${db.dbname}
db.delimiter=;
db.user=user
db.password=password
```

The file is referenced in the Ant build file as follows:

```
<property file="myprops.properties"/>
```

The following code excerpt loads the database data for app1 using the APP1DDL.sql file.

```
<target name="load-app1-tables">
  <antcall target="run-sql-script">
    <param name="db.acct" value="app1dbacct"/>
    <param name="db.script"
      value="${scripts.dir}/db/${app1dbacct.type}/APP1DDL.sql"/>
  </antcall>
</target>
```

The following code excerpt defines the run-sql-script target, which performs an SQL database call.

```
<target name="run-sql-script">
  <sql
    driver="${db.driver}"
    url="${db.sqltask.url}"
    delimiter="${db.delimiter}"
    userid="${db.user}"
    password="${db.password}"
    onerror="continue"
    autocommit="true"
    classpathref="basic.classpath"
    print="true"
    src="${db.script}"/>
</target>
```

Example: How to Create the Conversational State Database Tables Using Ant

The PO Sample on the dev2dev Web site provides an example of creating the conversational state database tables using Ant. You can access this article, and download its associated samples files at:<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=76>

A sample Java utility, `ManifestReader`, is included in the kit to read the `wlw-manifest.xml` file and extract the information necessary to create the conversational state database tables. This utility is referenced from a Jython script, `create_app_db_scripts`, which creates a DDL file with SQL commands to drop and create the conversational state database tables listed in the `wlw-manifest.xml` file.

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

Configuring the Production Database

Configuring Security

WebLogic Platform 8.1 provides a single, unified security framework for all deployments, including portal applications, integration applications, and J2EE applications running on WebLogic Server.

This section summarizes the ways that you can secure the target environment, provides considerations for configuring security, and describes how to promote embedded LDAP security data to the target database. This section also describes how to maintain security policy files under version control.

Topics include:

- [“Ways to Secure the Target Environment” on page 5-2](#)
- [“Considerations for Configuring Security” on page 5-6](#)
- [“Promoting Embedded LDAP Security Data to the Target Database” on page 5-7](#)
- [“Maintaining Security Policy Files Under Version Control” on page 5-8](#)

For more details about WebLogic security features, see the following resources:

- [Security in WebLogic Platform 8.1](#)
- [Introduction to WebLogic Security](#)

Note: BEA strongly recommends that you maintain an archive of your user information in the event of a machine failure or other problem. For more information, see “Backing Up User Information” in “Configuring WebLogic Platform Security” in [“Managing WebLogic Platform Security”](#) in *Security in WebLogic Platform 8.1*.

Ways to Secure the Target Environment

Table 5-1 lists ways you can secure a target environment, and includes recommendations for securing a production environment, in particular.

The following figure illustrates a WebLogic domain within a typical network configuration. Refer to this diagram when reviewing the security areas in Table 5-1.

Figure 5-1 WebLogic Domain Within a Typical Network Configuration

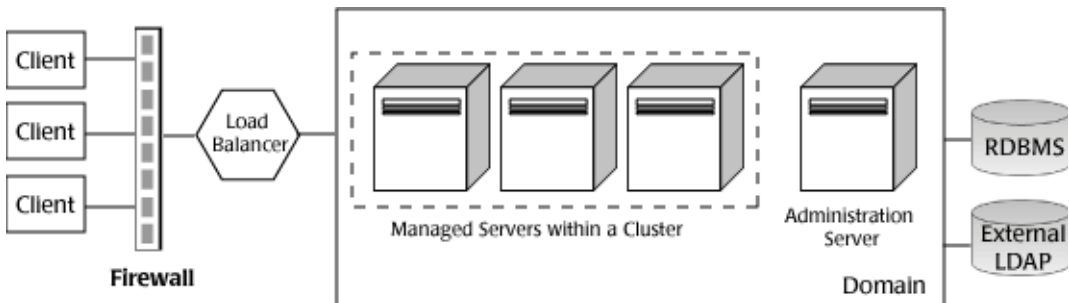


Table 5-1 Ways to Secure the Target Environment

Security Area	Considerations
Configure a firewall	<p>A firewall limits traffic between two networks. Firewalls can be a combination of software and hardware, including routers and dedicated gateway machines.</p> <p>Placing a firewall in front of your load balancing hardware enables you to set up a De-Militarized Zone (DMZ) for your web application using minimal firewall policies. The DMZ is a logical collection of hardware and services that is made available to outside, untrusted sources.</p> <p>For best practice tips when setting up a firewall, see “Firewall Considerations” in “Avoiding Problems” in “Clustering Best Practices” in <i>Using WebLogic Server Clusters</i>.</p>
Use a load balancer or Web proxy server	<p>A load balancer or Web proxy server distributes client connection requests, provides load balancing and failover across a WebLogic cluster, and provides security by concealing the local area network addresses from external users.</p> <p>For more information, see “Using Load Balancers and Web Proxy Servers” on page 6-1.</p>

Table 5-1 Ways to Secure the Target Environment (Continued)

Security Area	Considerations
Secure the network connections	When designing network connections, you balance the need for a security solution that is easy to manage with the need to protect strategic WebLogic resources. You need to determine whether to set up firewalls and connection filters, for example, to secure network access. For tips on securing the network connections, review “Securing Network Connections” in “ Ensuring the Security of Your Production Environment ” in <i>Securing a Production Environment</i> .
Secure the WebLogic Server hosts	A WebLogic Server production environment is only as secure as the security of the machine on which it is running. Therefore, it is important that you lock down the physical machine, the operating system, and all other software that is installed on the host machine. For tips on securing the WebLogic Server host, review “ Securing the WebLogic Server Host ” in <i>Securing a Production Environment</i> .
Secure the WebLogic domain	Install the Administration Server on its own machine and target applications on Managed Servers to insulate the application and its users from the production environment infrastructure. Do not target applications (other than WebLogic Portal applications and modules) on the Administration Server.
Secure WebLogic resources	<p>A <i>WebLogic resource</i> represents an underlying WebLogic Server entity that can be protected from unauthorized access. Examples of WebLogic resources include Enterprise Applications (EARs), EJBs (JARs), and Web Applications (WARs).</p> <p>A <i>security realm</i> comprises mechanisms for protecting WebLogic resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. A user must be defined in a security realm in order to access the WebLogic resources.</p> <p>For an overview of how resources are secured in WebLogic in a security realm, see “Security Realms” in <i>Introduction to WebLogic Security</i>. For information about customizing the default security realm, see “Customizing the Default Security Configuration” in <i>Managing WebLogic Security</i>.</p> <p>For more information about securing WebLogic resources, see:</p> <ul style="list-style-type: none"> • Securing WebLogic Resources • “Configuring Security Providers” in <i>Managing WebLogic Security</i>. • “Protecting Resources in a Platform-Based Application” in “Securing a Platform-Based Application” in “Introducing WebLogic Platform 8.1 Security” in <i>Securing in WebLogic Platform 8.1</i>. • “Protecting User Accounts” in <i>Managing WebLogic Security</i>.

Table 5-1 Ways to Secure the Target Environment (Continued)

Security Area	Considerations
Secure WebLogic Portal resources	<p>WebLogic Portal entitlements let you define rules (roles) to identify who can access specific portal resources. When a portal administrator logs in to the WebLogic Administration portal, the administrator sees only the areas that the administrator is allowed to administrator. When a visitor logs in to a portal, the visitor sees only the books, pages, and portlets to which the visitor is entitled.</p> <p>For more information about:</p> <ul style="list-style-type: none"> • The WebLogic Portal authentication framework, see “Securing Portal Applications” in <i>WebLogic Workshop Online Help</i>. • “Using Multiple Authentication Providers with WebLogic Portal” in <i>WebLogic Administration Portal Online Help</i>. • Configuring and promoting WebLogic Portal database tables, see “Creating and Preparing a Production Database” on page 4-1. • Using an external database store, see “Using an External Store for User Information” in <i>Security in WebLogic Platform 8.1</i>.
Secure WebLogic Integration resources	<p>You need to configure and manage security for trading partners for which security management is particularly important. Trading partners produce digital certificates for sending and receiving business messages in a secure environment.</p> <p>For more information about:</p> <ul style="list-style-type: none"> • WebLogic Integration security, see “Using WebLogic Integration Security” in <i>Deploying WebLogic Integration Solutions</i>. • Configuring and promoting WebLogic Integration database tables, see “Promoting WebLogic Integration Application Database Information” on page 4-5.

Table 5-1 Ways to Secure the Target Environment (Continued)

Security Area	Considerations
Secure the database	<p>A security provider database contains the users, groups, security roles, security policies, and credentials used by some types of security providers to provide security services. The security provider database can be the embedded LDAP server, a properties file, or a production-quality, customer-supplied database.</p> <p>The embedded LDAP server is the default security provider database for the WebLogic Authentication, Authorization, Credential Mapping and Role Mapping providers. The use of an external user store, such as an external LDAP or Relational Database Management System (RDBMS), enables centralized user management by security administrators across multiple domains.</p> <p>Also review the following information, as it applies to your environment:</p> <ul style="list-style-type: none"> • For tips on securing the database, review “Securing Your Database” in “Ensuring the Security of Your Production Environment” in <i>Securing a Production Environment</i>. • For information about managing the embedded LDAP server, see “Managing the Embedded LDAP Server” in <i>Managing WebLogic Security</i>. • If you are using an external user store, see “Using an External Store for User Information” in <i>Security in WebLogic Platform 8.1</i>.
Secure the applications	<p>Although most of the responsibility for securing the WebLogic resources in a WebLogic Server domain fall within the scope of the server, some security responsibilities lie within the scope of individual applications.</p> <p>For tips on securing the applications, review the following:</p> <ul style="list-style-type: none"> • “Preparing Application Security” on page 7-16 • “Securing Applications” in “Ensuring the Security of Your Production Environment” in <i>Securing a Production Environment</i> • Programming WebLogic Security
Enable Auditing	<p>BEA recommends that you enable auditing in a production environment. An auditing provider stores operating requests and the results of those requests are collected, stored, and distributed for the purposes of non-repudiation. For more information about configuring auditing, see “Configuring Security Providers” in <i>Managing WebLogic Security</i>.</p>

Considerations for Configuring Security

Consider the following when configuring security:

- Enable Secure Sockets Layer (SSL) security. By default, SSL is enabled and configured to use the demonstration Identity and Trust keystores. The demonstration keystores are suitable for a development environment, but should not be used in a production environment. You can configure SSL to use a commercial digital certificate (such as VeriSign). For information about configuring SSL, see [“Secure Sockets Layer \(SSL\)”](#) in *Introduction to WebLogic Security* and [“Configuring SSL”](#) in *Managing WebLogic Security*.
- Set the configuration startup mode to `prod`. For more information, see [“Configuring Servers to Start in Production Mode”](#) on page 3-14.
- Optionally, configure controls that receive external callbacks to enforce the role restrictions. Your application will run without these role restrictions, but it will not behave exactly as the developer intended in terms of the role restrictions placed on a control’s callback methods. This process is described in Step 3 of [“To Manually Create Required Resources on the Production Server”](#) in [“How Do I: Deploy a WebLogic Workshop Application to a Production Server?”](#) in the *WebLogic Workshop Online Help*.
- Enable trust between WebLogic Server domain, if required. BEA Systems does not recommend enabling trust between WebLogic Server domains in a production environment unless the servers have a secure means of communication such as a dedicated line or are protected by a firewall. For information, see [“Enabling Trust Between WebLogic Server Domains”](#) in *Managing WebLogic Security*.
- Do not configure the domain-wide Administration Port, as it is not supported for use by WebLogic Platform applications. You should use the SSL port. For more information about the Administration Port, see [“Enabling the Domain-Wide Administration Port”](#) in *WebLogic Server Administration Console Online Help*.
- If you wish to configure single sign-on, be careful that you do not inadvertently enable access permissions to untrusted users. For information about configuring single sign-on, see [“Single Sign-On with Enterprise Information Systems”](#) in *Managing WebLogic Security*.

Promoting Embedded LDAP Security Data to the Target Database

The following sections describe how you can promote existing embedded LDAP data to the target embedded LDAP database:

- [“Promoting Security Data to the Target Environment” on page 5-7](#)
- [“Promoting WebLogic Portal and WebLogic Integration Data” on page 5-8](#)

Note: WebLogic Platform does not support an automated process for promoting embedded LDAP server data to an external user store, such as an external LDAP server or RDBMS. If you develop a custom tool to handle this process, you should be aware that the existing password information is not maintained during the promotion.

Promoting Security Data to the Target Environment

The embedded LDAP server is the default security provider database for the WebLogic Authentication, Authorization, Credential Mapping and Role Mapping providers. The embedded LDAP server contains user, group, group membership, security role, security policy, and credential map information. By default, each WebLogic Server domain has an embedded LDAP server configured with the default values set for each attribute.

If you have created security information, or configured security providers, in development that you expect to be used in the target environment, you will want to promote that information and security provider configuration to the target environment. Promoting this data ensures that your application will work correctly in the target environment. WebLogic Server provides utilities you can use to export the following security data from one security realm, and import them into a new security realm:

- Users and groups
- Security policies
- Security roles
- Credential maps

You can migrate security data for each security provider individually, or migrate security data for all the WebLogic security providers for an entire security realm at once. You migrate security data through the WebLogic Server Administration Console or by using the `weblogic.admin` utility. For information about importing and exporting security data from security realms and security providers, see:

- [“Migrating Security Data”](#) in *Managing WebLogic Security*
- [“Exporting and Importing Information in the Embedded LDAP Server”](#) in *Managing WebLogic Security*

Note: Optionally, you can use an LDAP browser to export and import data stored in the embedded LDAP Server.

Promoting WebLogic Portal and WebLogic Integration Data

When you set up the production database, you must define the database tables required by your application. In some cases, you can promote existing security data to the target environment. For information about configuring and promoting WebLogic Portal and WebLogic Integration data, see [“Promoting Database Information to the Production Database”](#) on page 4-3.

Maintaining Security Policy Files Under Version Control

You can maintain the security policy files using a source control tool, such as Perforce or CVS, by performing the following steps:

1. Export security policy information, as described in the following sections:
 - [“Migrating Security Data”](#) in *Managing WebLogic Security*
 - [“Exporting and Importing Information in the Embedded LDAP Server”](#) in *Managing WebLogic Security*
2. Check the data into your version control system.

Using Load Balancers and Web Proxy Servers

In a production environment, a *load balancer* or *Web proxy server*, as illustrated in the “[WebLogic Platform Domain Examples](#)” on page 2-6, is used to distribute client connection requests, provide load balancing and failover across the cluster, and provide security by concealing the local area network addresses from external users. A load balancer or Web proxy server allows all applications in the domain to be represented as a single address to external clients, and is required when using in-memory replication for client session information.

This section describes how to configure a hardware load balancer or Web proxy server.

Topics include:

- “[Load Balancing with an External Hardware Load Balancer](#)” on page 6-1
- “[Load Balancing with a Web Proxy Server](#)” on page 6-2
- “[Considerations When Configuring Load Balancers and Web Proxy Servers](#)” on page 6-3

Load Balancing with an External Hardware Load Balancer

If you are using load balancing hardware, instead of a proxy, it must support a compatible passive or active cookie persistence mechanism, and SSL persistence, as described below:

- Passive cookie persistence enables WebLogic Server to write a cookie containing session parameter information through the load balancer to the client. When using some hardware load balancers, you must configure the passive cookie persistence mechanism to avoid overwriting the WebLogic Server cookie that tracks primary and secondary servers used for in-memory replication. Specifically, you must set the following values:

- String offset value to the Session ID value plus 1 byte for the delimiter character
- String length to 10 bytes
- Active cookie persistence is supported as long as the mechanism does not overwrite or modify the WebLogic HTTP session cookie. In this case, no additional configuration is required.
- SSL persistence performs all encryption and decryption of data between clients and the cluster, and uses a plain text cookie on the client to maintain an association between the client and a server in the cluster.

For more information about active and passive cookie persistence and SSL persistence, see:

- “Load Balancing HTTP Sessions with an External Load Balancer” in “Load Balancing for Servlets and JSPs” in [“Load Balancing in a Cluster”](#) in *Using WebLogic Server Clusters*.
- “Configuring Load Balancers that Support Passive Cookie Persistence” in “Cluster Implementation Procedures” in [“Setting Up WebLogic Clusters”](#) in *Using WebLogic Server Clusters*.

For a description of connection and failover for HTTP session in a cluster with load balancing hardware, see “Accessing Clustered Servlets and JSPs with Load Balancing Hardware” in “Replication and Failover for Servlets and JSPs” in [“Failover and Replication in a Cluster”](#) in *Using WebLogic Server Clusters*.

If you are using an F5 BIG-IP hardware load balancer, see also [“Configuring BIG-IP Hardware with Clusters”](#) in *Using WebLogic Server Clusters*.

Load Balancing with a Web Proxy Server

A Web proxy server maintains a list of WebLogic Server instances that host a clustered servlet or JSP, and forwards HTTP requests to those instances. Requests are forwarded on a round-robin basis, by default, as described in “Round Robin Load Balancing” in “Load Balancing for EJBs and RMI Objects” in [“Load Balancing a Cluster”](#) in *Using WebLogic Server Clusters*.

You can implement a Web proxy server using WebLogic Server with the `HttpClusterServlet` or by using one of the following Web servers and associated proxy plug-ins:

- Netscape Enterprise Server with the Netscape (proxy) plug-in
- Apache with the Apache Server (proxy) plug-in
- Microsoft Internet Information Server with the Microsoft IIS (proxy) plug-in

Refer to the following table for information about configuring a Web proxy server to load balance servlets and JSPs.

Table 6-1 Configuring a Web Proxy Server

If your Web server is...	Then set up...
WebLogic Server	The <code>HttpClusterServlet</code> as a Web proxy server, as described in: <ul style="list-style-type: none"> • “Creating HTTP Proxy Applications” in “Configuring Managed Servers, Clusters, and Machines” in <i>Creating WebLogic Domains Using the Configuration Wizard</i> • “Configuring Proxy Plug-Ins” in <i>Using WebLogic Server Clusters</i>
A third-party Web server	The corresponding plug-in, as described in Using WebLogic Server Plug-ins with WebLogic Server .

Considerations When Configuring Load Balancers and Web Proxy Servers

The following lists considerations for configuring load balancers and Web proxy servers.

- There may be circumstances in which you have to configure the WebLogic Workshop runtime to work with a Web server proxy. For more information, see “Proxy Server Setup” in [“Clustering Workshop Applications”](#) in the *WebLogic Workshop Online Help*.
- To support automatic failover, WebLogic Server replicates the HTTP session state of clients that access clustered servlets and JSPs, and maintains them in memory, a file system, or a database. To utilize in-memory replication for HTTP session states, you must access the WebLogic Server cluster using either a collection of Web servers with identically configured WebLogic proxy servers or load balancing hardware.

The primary session state is stored on the server to which the client first connects. By default, WebLogic Server attempts to create session state replicas on a different machine than the one that hosts the primary session state. You can control where secondary states are placed using replication groups. A *replication group* is a preferred list of clustered servers to be used for storing session state replicas. For more information, see:

- Enabling session replication in your applications, see [“Enabling Session Replication” on page 7-14](#).
- Defining replication groups, see [“Using Replication Groups”](#) in *Using WebLogic Server Clusters*.

- HTTP session replication and how clustered servlets and JSP are accessed using load balancers, see “Replication and Failover for Servlets and JSPs” in “[Failover and Replication in a Cluster](#)” in *Using WebLogic Server Clusters*.
- When using T3 tunneling with a load balancer, you should set the load balancing algorithm to one of the server affinity algorithms to ensure that clients connect only through the tunneled connection. For more information, see “Round-Robin Affinity, Weight-Based Affinity, and Random Affinity” in “Load Balancing for Servlets and JSPs” in “[Load Balancing in a Cluster](#)” in *Using WebLogic Server Clusters*.
- In addition to distributing HTTP traffic, external load balancers can distribute initial context requests that come from Java clients over T3 and the default channel. For a discussion of object-level load balancing, see “Load Balancing for EJBs and RMI Objects” in “[Load Balancing in a Cluster](#)” in *Using WebLogic Server Clusters*.

Preparing the Application for Deployment

As the application moves through the promotion stages, application source and configuration files may need to be modified before the application can be deployed. Modifications are necessary if the new target environment is different from the previous one in the following ways:

- The application will be used in a high-availability environment. For example, being moved from a non-clustered to a clustered domain.
- Web services or other applications accessed by the application are no longer located on the same server, or the same domain.
- Security features are being added to the application, such as restricting access to authenticated users; or the application is restricted to receiving only SSL requests.

This chapter explains specific changes that you may need to make, the circumstances in which you might need to make those changes, and how to make them. The following topics are included:

- [Summary of Application Preparation Tasks](#)
- [Summary of Changes to Deployment Descriptors and Configuration Files](#)
- [Managing Application Files for Promotion to Production](#)
- [Enabling High Availability](#)
- [Reconfiguring Application Views and Adapters](#)
- [Preparing Application Security](#)
- [Packaging the Application](#)

Summary of Application Preparation Tasks

Table 7-1 lists tasks that you may need to perform for application files depending on the characteristics of the application, on the target environment, or both, and provides links to topics that explain how to do those tasks.

Table 7-1 List of Application Preparation Tasks You Might Need to Perform

Application or Target Environment Characteristics	Tasks You May Need to Perform	For more information, see . . .
Any post-development environment	Obtain application files.	“Access to Application Files” on page 7-7
	Update application project files to point to the domain in the target environment on which the application runs.	“Updating the Server Path Attribute” on page 7-18
	Update WebLogic Portal cache configuration, behavior tracking, campaign, and commerce tax settings.	Instructions about updating the application-config.xml file in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3
	Tune the application for a production environment.	Instructions for modifying the <jsp-descriptor> element of the weblogic.xml deployment descriptor in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3.
	Tune WebLogic Workshop dispatcher and container EJBs for a production environment.	Instructions for modifying these EJBs in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3.
The application is targeted to a cluster.	Enable session replication to maintain information about object state in the event of a server failure.	“Enabling Session Replication” on page 7-14

Table 7-1 List of Application Preparation Tasks You Might Need to Perform (Continued)

Application or Target Environment Characteristics	Tasks You May Need to Perform	For more information, see . . .
The application uses Web service controls to access other applications.	Update host, port, and optionally security protocol of target service specified in the Web service controls.	“Enabling Inter-Application Communication” on page 7-11
The application uses the Business Process Monitoring features of WebLogic Integration.	Set the concurrency strategy for stateful business process entity beans.	“Specifying a Concurrency Strategy for Stateful Business Process Entity Beans” on page 7-14
The application uses Application Integration features of WebLogic Integration.	Update the WebLogic Integration application views.	“Reconfiguring Application Views” on page 7-15
	Specify Application Integration adapter target event generators.	“Specifying Event Generator Targets” on page 7-16
Secure access to the application is needed.	Restrict application access to authenticated users only.	“Limiting Access to Authenticated Users” on page 7-16
	Restrict application access to SSL requests only.	“Restricting Application Access to SSL Traffic Only” on page 7-17

Summary of Changes to Deployment Descriptors and Configuration Files

Like any J2EE application, a WebLogic Platform application has a number of deployment descriptors that you might need to tune for your production environment. The production environment also has a number of properties files that you might need to modify as well. The following table identifies these files, summarizes the changes that may be needed, identifies how to make the changes and when they can take effect, and provides links to topics that provide more information. Each file described in the following table, and the directory in which it is located, is shown in **bold** in the first column. In this table:

- *<App-dir>* represents the root directory of the application.

Preparing the Application for Deployment

- `<Web-project>` represents the top-level directory of a Workshop project in the application.
- `<Domain-root>` represents the root directory of the domain.

Table 7-2 Summary of Changes that May Be Needed to Deployment Descriptors and Configuration Files

File and Location	Description
<pre><App-dir> APP-INF META-INF application-config.xml</pre>	<p>Description:</p> <p>WebLogic Portal application configuration file. Contains cache configuration, behavior tracking, campaign, and commerce tax settings.</p> <p>When changes might be necessary:</p> <p>If these values are different in the new target environment, modify this file appropriately in a text editor before building the portal application.</p> <p>When changes take effect:</p> <p>After the application is rebuilt.</p> <p>For more information, see:</p> <p>“Preparing and Deploying the EAR File” in the WebLogic Portal <i>Production Operations User Guide</i> at http://e-docs.bea.com/wlp/docs81/prodOps/deployment.html</p>
<pre><App-dir> APP-INF META-INF <Web-project> WEB-INF web.xml</pre>	<p>Description:</p> <p>J2EE Web application deployment descriptor.</p> <p>When changes might be necessary:</p> <p>If the application is to be restricted to receiving SSL requests, you may need to specify the <code>CONFIDENTIAL</code> parameter in the <code><user-data-constraint></code> element.</p> <p>When changes take effect:</p> <p>If you make changes to this deployment descriptor, changes take effect when the application is redeployed.</p> <p>For more information, see:</p> <p>“Restricting Application Access to SSL Traffic Only” on page 7-17</p>

Table 7-2 Summary of Changes that May Be Needed to Deployment Descriptors and Configuration Files

File and Location	Description
<pre> <App-dir> APP-INF META-INF <web-project> WEB-INF weblogic.xml </pre>	<p>Description:</p> <p>The standard WebLogic deployment descriptor for Web applications.</p> <p>When changes might be necessary:</p> <p>Applications targeted to a cluster should have session replication enabled, which you configure via the <code><session-param></code> descriptor element using a text editor. We also recommend the following changes to the <code><jsp-descriptor></code> element to tune the application appropriately for a production environment:</p> <ul style="list-style-type: none"> • Turn off debugging by setting <code>debug</code> to <code>false</code>. • Precompile the JSPs in the Web application to reduce the time needed to display pages on their first invocation by setting <code>precompile</code> and <code>preCompileContinue</code> to <code>true</code>. • Disable polling of JSP pages for changes by setting <code>pageCheckSeconds</code> to <code>-1</code>. <p>When changes take effect:</p> <p>Changes to this file require redeployment of the application.</p> <p>For more information, see:</p> <p>For enabling session replication, see “Enabling Session Replication” on page 7-14.</p>
<p>Deployment descriptors for the following EJBs created in WebLogic Workshop:</p> <ul style="list-style-type: none"> • Dispatcher EJBs • Container EJBs 	<p>Description:</p> <p>Additional deployment descriptors for Web services created in WebLogic Workshop.</p> <p>When changes might be necessary:</p> <p>You may need to modify these deployment descriptors if those Web services are being moved to a production environment.</p> <p>When changes take effect:</p> <p>Changes to these files require redeployment of the application.</p> <p>For more information, see:</p> <p>Information is available on dev2dev in the <i>WebLogic Workshop Internals</i> document, in the topic “Application Customization,” at the following URL: http://dev2dev.bea.com/products/wlworkshop81/articles/wl_internals.jsp#65</p>

Table 7-2 Summary of Changes that May Be Needed to Deployment Descriptors and Configuration Files

File and Location	Description
<pre><Domain-root> jws-config.properties</pre>	<p>Description:</p> <p>Properties file that defines resources used by the WebLogic Workshop run-time framework, such as the data source used to persist conversational state and the JMS server used to buffer asynchronous service requests.</p> <p>When changes might be necessary:</p> <p>If the location or configuration of these resources changes in the new target environment, updates to this properties file will be necessary.</p> <p>When changes take effect:</p> <p>Changes to this properties file take effect on server restart.</p> <p>For more information, see:</p> <p>“jws-config.properties Configuration File” in the <i>WebLogic Workshop Help</i> at the following URL:</p> <p>http://edocs.bea.com/workshop/docs81/doc/en/workshop/reference/configfiles/con_jws-config_properties_ConfigurationFile.html</p>
<pre><App-dir> APP-INF META-INF <Web-project> WEB-INF wlv-config.xml</pre>	<p>Description:</p> <p>Web application project configuration file. When the application is built into an EAR file for deployment, the values in this file override the application’s default values.</p> <p>When changes might be necessary:</p> <p>Specifying values in this file is useful for setting the concurrency strategy for stateful business process entity beans.</p> <p>When changes take effect:</p> <p>If you modify this file, you must rebuild and redeploy the application.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • “Specifying a Concurrency Strategy for Stateful Business Process Entity Beans” on page 7-14 • “wlv-config.xml Configuration File” in the <i>WebLogic Workshop Help</i> at the following URL: <p>http://e-docs.bea.com/workshop/docs81/doc/en/workshop/reference/configfiles/con_wlv-config_xml_ConfigurationFile.html</p>

Table 7-2 Summary of Changes that May Be Needed to Deployment Descriptors and Configuration Files

File and Location	Description
<code><Domain-root></code> <code>wli-config.properties</code>	<p>Description:</p> <p>Properties file that defines resources used by the WebLogic Integration run-time framework, such as the document size threshold for using the document store.</p> <p>When changes might be necessary:</p> <p>If the location or configuration of these resources changes in the new target environment, updates to this properties file will be necessary.</p> <p>When changes take effect:</p> <p>If you make modifications, they take effect only after a restart of the WebLogic Server instance on which the application is deployed.</p> <p>For more information, see:</p> <p>“wli-config.properties Configuration File” in <i>Deploying WebLogic Integration Solutions</i> at the following URL:</p> <p>http://edocs/wli/docs81/deploy/wliconfig_appx.html</p>

Managing Application Files for Promotion to Production

This section contains the following topics, which make recommendations for managing application files for promotion to production:

- [Access to Application Files](#)
- [Using a Version Control System](#)
- [Files that Should Not Be Under Version Control](#)

Access to Application Files

The two primary mechanisms for providing access by an application administrator to application files include the following:

- The files are accessed from a version control system, such as the Revision Control System (RCS) or Perforce
- An application administrator is given an application EAR file

Two considerations for choosing a mechanism include:

- The role of the application administrator, particularly whether he or she should be responsible for making source code changes
- The method chosen for making source file changes: in the build environment, or by editing deployment descriptors directly

One main advantage of using a version control system is that changes to application files can be tracked, and such a system is usually integrated with the WebLogic Workshop IDE. This integration provides a convenient means to rebuild application files. However, changes made to those files for the purposes of deploying into a target environment need to be tracked. In larger team environments, complications may arise if changes are made simultaneously by multiple team members.

However, when an application administrator is given an application EAR file, access to the build environment might not be available. If a change requires a rebuild of the application files, the application administrator might not be able to make the change, which could be a limitation. But when the method chosen for making file changes is to edit deployment descriptors directly, nothing more than access to the EAR file is needed. Deployment descriptors can be easily extracted, modified, and staged. In addition, handing off the EAR file to the application administrator might work better to control changes made to application source files. Moreover, it is usually easier to create scripts that work directly with EAR files. Depending on the policies that an IT team uses, the role of the application administrator, the promotion process, and the application, one mechanism might be more appropriate than the other.

Most of the changes described in this chapter are made to the deployment descriptors, which are automatically available in the EAR file, and not to application source files. So the issue of source file control, unless otherwise noted, is not a factor regarding this chapter's scope.

Using a Version Control System

It is highly recommended to use a source control tool for version control and maintain an archive of application files. WebLogic Workshop integrates directly with CVS, Perforce, and IBM Rational ClearCase. Once you have added the files in your WebLogic Workshop application to a repository managed by one of these source control products, you can check files in and out using commands available in WebLogic Workshop.

When developing an application in WebLogic Workshop, the files to add to source control are:

- The `.work` file that represents the application. It appears at the root of the application directory.

- All source files that you have added or modified—Web services, Java controls, JSPs, business processes, and so on.
- Any XML schema files that you have added to the Schemas project.
- The files in the `resources` directory in a Web project.
- Any JAR files that you have added to the `modules` directory. These files are stored at the root of your application in the file system.
- Any JAR files that you have added to the libraries folder. These files are stored in the `APP-INF/lib` directory in the file system.
- The following files in the `WEB-INF` folder in a Web project:
 - `web.xml`
 - `weblogic.xml`
 - `wlw-config.xml`
- The `global.app` file in the `WEB-INF/src/global` directory.
- The tag libraries that appear in the `WEB-INF` directory in a Web project, if you have page flows and JSP files in your project. These files end with the `.tld` and `.tldx` extensions.
- The JAR files that appear in the `WEB-INF/lib` directory in a Web project.

For more information about integrating a version control system with WebLogic Workshop projects, see “Integrating with Source Control Systems” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/guide/devenv/conIntegratingWithSourceControlSystems.html>

Files that Should Not Be Under Version Control

While it is important to use a source control tool, such as Perforce or CVS, there are files that should not be checked into the source tree managed by such a tool. These include files that need to be installed before building the application for production deployment. These include JAR and library files provided by WebLogic Workshop for WebLogic Portal and WebLogic Integration, and also include files such as NetUI tag libraries and Liquid Data Control files. These files are staged by WebLogic Workshop automatically when you enable the templates for the applications you create for those products.

In addition, you should not check in files that are automatically generated by WebLogic Workshop during development, which include the following:

```
.workshop directory
META-INF (except the application-config.xml file)
<web-project>/WEB-INF/*.tld
<web-project>/WEB-INF/*.tldx
<web-project>/WEB-INF/jpf-struts-config*.xml
<web-project>/WEB-INF/classes
<web-project>/WEB-INF/lib
<schema-project>/system/*. *
*.jar
*.war
```

In the preceding list:

- *<web-project>* represents the top-level directory name for your application's WebLogic Workshop project in the domain directory; for example:

```
c:\bea\user_projects\domains\workshop\MyPlat-web-project
```

- *<schema-project>* represents the top-level directory of the optional schema project, which contains your application's schema (XSD) files.

Note: If a JAR file was generated in a different application and you add it as a library or a module to the current application, put it under source control. However, if you build the library in your application, do not put it under source control, and do not copy it into the production domain. Remember that JAR files added as a module in the **Modules** folder of WebLogic Workshop's **Application** pane are placed by WebLogic Workshop in the root folder of your application, while JARs added as a library in the **Libraries** folder of WebLogic Workshop are placed in the `APP-INF/lib` directory.

Enabling High Availability

When developing an application in WebLogic Workshop, you typically do not need to be concerned with several aspects of the production environment, such as:

- Access to resources in a distributed, heterogeneous environment
- High availability considerations, such as clustering and load balancing

The strength of WebLogic Workshop is that it allows you to focus on the business logic of your application, freeing you from having to write code that enables the portability, reliability, availability, and scalability features that typically are required for an enterprise-level application

in a production environment. WebLogic Workshop does this, in part, by making a number of resources and capabilities available automatically, or in a very simplified way, in the development environment; for example, database access, client access, and communication with other applications.

When you move an application from development to production, however, you need to make a small number of modifications to application files to enable the application to access resources, and be accessible by clients, in a clustered environment. By making these simple modifications, you enable your application to fully leverage the high-availability and load-balancing capabilities provided by a clustered WebLogic domain environment.

There are three key areas where you typically need to make modifications so your application can function in a clustered production domain:

- [Enabling Inter-Application Communication](#)
- [Enabling Session Replication](#)
- [Specifying a Concurrency Strategy for Stateful Business Process Entity Beans](#)

Enabling Inter-Application Communication

When building applications that must interoperate with distributed systems over a network, using Web service controls can be a powerful and convenient means to enable communication with back-end resources and other applications. In a development environment, applications can use *direct access* to invoke a Web service or Web application—for example, by specifying `localhost:7001` as the service's host and port. However, in a production environment, those Web services typically are not available locally, or their locations may have changed, or they may be accessed only via a Web proxy server or a load balancer to servers in the target environment.

Therefore, one challenge of moving an application from the WebLogic Workshop development environment to a production environment is ensuring that Web service controls used in that application contain correctly updated URLs. The information presented in this section does not simply explain a procedural step for application deployment, but also provides important considerations for the proper design of the application.

The need to update URLs in Web service controls typically occurs in the following situations:

- The target Web service and the application that invokes the Web service are being developed simultaneously, and the host and port of the target Web service in the production environment is either not established or subject to change.

- The target Web service is currently deployed in the production environment, but the application under development must use a staged version of that Web service until the application is also in production.

In either situation, the host and port specified in the Web service control is subject to change until the application is deployed in the production environment. This section shows an example Web service control URL, and describes four techniques you can use to update that URL.

Example Web Service Control URL

When you incorporate a Web service control in an application to invoke a Web service, WebLogic Workshop generates the host and port of the invoked Web service into a JCX file, and includes the host and port as part of the `@jc:location` annotation. The following example shows the `@jc:location` annotation for the Web service control generated during development. The host and port are shown in **bold**.

```
/**
 * @jc:location http?url="http://localhost:7001/processes/CSR.jspd"
 * @jc:wSDL file="#CSR_ProcessWSDL"
 */
```

Techniques for Updating URLs in Web Service Controls

Before you build the application EAR file for production deployment, the host and port in the `@jc:location` annotation must be updated to specify the Web proxy server or load balancer of the production domain in which the target Web service is deployed, as in the following example.

```
/**
 * @jc:location http?url="http://proxysvr2:9201/processes/CSR.jspd"
 * @jc:wSDL file="#CSR_ProcessWSDL"
 */
```

You can use the following techniques to update host and port information specified for Web service controls in your application:

- [Technique 1: Reimport the target Web service's WSDL file into your project](#)
- [Technique 2: Manually update your application's @jc:location annotations](#)
- [Technique 3: Implement an invocation to the setEndPoint\(\) method on the service control](#)
- [Technique 4: Implement a Service Broker control](#)

Techniques 1 and 2 involve source-level changes to the application, so access to the build environment is required to make these changes, and the application must be rebuilt for the changes to take effect. Techniques 3 and 4 do not require source-level changes to application files at deployment time as the previous two techniques do, but they do reflect features that would need to have been designed into the application during development.

Regardless of which technique you use, the development team needs to provide a list of remote services that are used by the application so that production environment system administrator can update the corresponding service control files appropriately before doing the final production environment build and deployment.

Technique 1: Reimport the target Web service's WSDL file into your project

When you move your application into a production domain, you can reimport the target Web service's WSDL file into your project. This WSDL file needs to have been generated from the target Web services after they have been deployed in the production environment. This method ensures an accurate update of your application's Web service controls.

Technique 2: Manually update your application's `@jc:location` annotations

In WebLogic Workshop, you can manually update each `@jc:location` to specify the correct host and port for each target Web service. This method may be quickest and simplest, but this also requires a recompilation of your application in the IDE. Because you are doing the updates manually, there is the risk of errors.

Technique 3: Implement an invocation to the `setEndPoint()` method on the service control

Each service control has a `setEndPoint()` method that you can implement to force the control, at run time, to use URL endpoint information included in a properties file that you create.

Considerations for using this technique include:

- Determining where and how you obtain the endpoint information, which can be specified in a separate file, and ensuring that the information is correct and available in each target environment in which the application is deployed.
- Making this endpoint information available to the application at run time in an efficient way; for example, by not forcing the application to obtain the endpoint information more than once.
- Updating the endpoint information if the production deployment changes.

Technique 4: Implement a Service Broker control

The Service Broker control allows a business process to send requests to and receive callbacks from another business process, a Web service, or a remote Web service or business process. The Service Broker control is an extension of the Web service control, and it allows you to pass endpoint information to your application dynamically. For example, you can specify properties in the WebLogic Integration Administration Console. Implementing this method does not require a rebuild of the application.

Enabling Session Replication

WebLogic Server uses standards-based communication techniques and facilities to share and maintain information about the availability of objects in a cluster. These techniques allow WebLogic Server to determine whether an object has stopped before finishing its job (because the machine on which the application is running has failed), and where there is a copy of the object to complete the job that was interrupted.

Information about what has been done on a job is called state. One technique used by WebLogic Server to maintain information about state is session replication. When a particular object unexpectedly stops doing its job, session replication enables a copy of the object to pick up where the failed object stopped, and finish the job.

Applications that are deployed in a WebLogic Platform domain should have session replication enabled. Without this setting, you will not have failover of a user's state information if a server in the cluster is stopped. To enable session replication across the cluster, configure the `<session-param>` descriptor element in the `weblogic.xml` deployment descriptor file for the application, as in the following example:

```
<session-descriptor>
  <session-param>
    <param-name>PersistentStoreType</param-name>
    <param-value>replicated_if_clustered</param-value>
  </session-param>
</session-descriptor>
```

Specifying a Concurrency Strategy for Stateful Business Process Entity Beans

Most database systems, such as Oracle, do not hold a lock after an `insert` operation. The first transaction of a business process always includes an `insert` operation. If a business process is

intended to receive another message or callback before the first transaction completes, database concurrency may be insufficient. In this case, use exclusive concurrency. Note that the subsequent messages or callbacks must originate on the local server for this to work.

To enable exclusive concurrency, make sure that the `<ejb-concurrency-strategy>` element in the `wlw-config.xml` file is set to `exclusive` for stateful business process entity beans. For information about issues regarding specific databases, see the *BEA WebLogic Integration Release Notes* at the following URL:

<http://e-docs.bea.com/wli/docs81/relnotes/index.html>

For information about modifying the `wlw-config.xml` file, see “`wlw-config.xml` Configuration File” in the *WebLogic Workshop Help* at the following URL:

http://e-docs.bea.com/workshop/docs81/doc/en/workshop/reference/configfiles/con_wlw-config_xml_ConfigurationFile.html

Reconfiguring Application Views and Adapters

The `aiConfigurator` utility allows an administrator to modify environment-specific information across application view, adapter, and connection factory descriptors used with WebLogic Integration. This allows an administrator to preconfigure application integration resources to deploy correctly in a production system environment. Further tuning of these resources can then be performed using the WebLogic Integration Administration Console.

The following sections explain and provide examples of how you can use the `aiConfigurator` utility to reconfigure application views and target event generators:

- [Reconfiguring Application Views](#)
- [Specifying Event Generator Targets](#)

Reconfiguring Application Views

If your application uses application views from WebLogic Integration as interfaces to services and events in an application enabled by one or more adapters, those applications views must be updated as appropriate for the configuration in the production environment. For example, if your application uses an application view with a DBMS adapter, the settings for the DBMS used in the production domain will be different from those in the development domain.

BEA provides the `aiConfigurator` utility to automate the reconfiguration of application views. For information about using the `aiConfigurator` utility, see “Administering Environment-Specific Application Integration Information” in *Deploying WebLogic Integration Solutions* at the following URL:

http://e-docs.bea.com/wli/docs81/deploy/config_appx.html

Specifying Event Generator Targets

When using an Application Integration adapter that supports multiple generator instances, such as the sample DBMS adapter, event generators can be distributed to each node in the cluster. This is done by specifying the Event Generation Targets using the `-inboundMessagingTargets` option to the `aiConfigurator` utility.

The target specification depends on whether the adapter has implemented *event generator instance support*. Considerations for implementing event generator support are described in “Developing an Event Adapter” in *Developing Adapters* at the following URL:

<http://e-docs.bea.com/wli/docs81/devadapt/7devea.html>

Preparing Application Security

The following sections explain changes that may need to be made to application files depending on the security requirements of the target environment:

- [Limiting Access to Authenticated Users](#)
- [Restricting Application Access to SSL Traffic Only](#)

Limiting Access to Authenticated Users

If access to the application in the target environment is to be restricted only to authenticated users, you need to specify either `FORM` or `CLIENT-CERT`, as appropriate, in the `<login-config>` element of the `web.xml` deployment descriptor.

- Specify `FORM` to use form authentication, which requires a user to provide a valid username and password to use the application. The user must be a member of an appropriate role before the user can access the application.
- Specify `CLIENT-CERT` to use client certificate authentication, which requires the use of client certificates to enable access to the application.

The following example shows setting the `<auth-method>` subelement of the `<login-config>` element to `CLIENT-CERT`:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

Restricting Application Access to SSL Traffic Only

If the security model for the target environment is to restrict the application to receive SSL requests only, regardless of whether the host server is configured to receive requests only on an SSL listening port, complete the following steps.

1. Update Web Service controls for the invoking application

“[Enabling Inter-Application Communication](#)” on page 7-11 explains how to update Web service controls to specify the host and port of the Web proxy or load balancer that handles access to target Web services invoked by your application. If you want to restrict application access to SSL requests only, you need to update the same Web service controls to specify a secure transport in the Web service URLs.

The following code snippet shows the secure transport that needs to be specified in the target Web service URL in a service control:

```
/**
 * @jc:location http-url="https://proxysvr1:7002/webservice/Report.jws"
 * @jc:wSDL file="#ReportWSDL"
 */
public interface ReportControl extends
com.bea.control.ControlExtension, com.bea.control.ServiceControl
```

This update can be made at the same time you modify the host and port in the target Web service URL, as explained in “[Techniques for Updating URLs in Web Service Controls](#)” on page 7-12.

Note: The application also needs to have additional mechanisms implemented in its code in order to use one or two-way SSL; however, this is a development task, not deployment. For more information, see “WebLogic Workshop Security Overview” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/guide/security/navSecurity.html>

2. For an application to receive only requests delivered via SSL, specify `CONFIDENTIAL` in the `<transport-guarantee>` section of the `<user-data-constraint>` element in the application’s `web.xml` deployment descriptor, located in the `WEB-INF` directory. For example:

```
<security-constraint>
  <!--some other configs -->
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
```

```
</user-data-constraint>  
</security-constraint>
```

Warning: Domain-level security also needs to be set up properly; namely, the SSL listen port has to be configured. Corresponding changes to the invoking application are needed to enable the use of SSL. For an example of sending a client certificate, see “WebServiceA.jws Sample” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/samplesrc/SamplesApp/WebServices/security/transport/clientCert/WebServiceA-jws.html>

Packaging the Application

WebLogic Workshop produces J2EE enterprise applications for deployment in a production domain. In an archived format, these are typically EAR files. Note that you cannot deploy a Web application project alone; you can deploy a Web application project only as part of an entire application.

The following topics are included:

- [Updating the Server Path Attribute](#)
- [Generating the EAR File](#)

Updating the Server Path Attribute

When you move an application from one environment to another, which you do at each stage of promotion, the application EAR file must be rebuilt via the `wlwBuild` command. When an application’s projects are moved to another environment, you must reset the `server.path` attribute of the application’s `.work` file to point to the target environment’s root domain directory.

The application’s `.work` file is located in the root of the application directory.

In the following example, the `server.path` attribute is set to the domain `platDomain`:

```
<option name="server.path" value="../../../domains/platDomain"/>
```

For more information about the `wlwBuild` command, see “`wlwBuild` Command” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/reference/commands/cmdWlwBuild.html>

Generating the EAR File

You can generate an EAR file for a WebLogic Workshop application using either of the following methods:

- In the IDE, by selecting **Build EAR** from the **Build** menu
- Using the `wlwBuild` command line tool

The `wlwBuild` tool is somewhat more flexible because you can set flags to build a JAR file for a specific project, instead of building an EAR file for the entire application. In addition, the `wlwBuild` tool can be easily invoked from an Ant task that you can either autogenerate or create manually. For details, see “How Do I: Call `wlwBuild.cmd` from an Ant build.xml file?” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/guide/deployment/howCallWLWBuildFromAnt.html>

When building an application, note the following:

- Before packaging a Portal application into an EAR file, use the WebLogic Administration Portal to create any content management repositories you want to use in your application. For details, see “Add a New Repository Connection” in the *WebLogic Administration Portal Help* at the following URL:

http://e-docs.bea.com/wlp/docs81/adminportal/help/CM_CreateNewRepository.html

- If you are going to use a particular Java Virtual Machine (JVM) in your production environment, and that JVM is supported in the IDE on the development machine you are using, it is a good idea to compile the EAR application with the SDK for that JVM. You can change the JVM for your WebLogic Workshop project by going to **Tools**→**Application Properties**, selecting **WebLogic Server**, and specifying the path to the JDK Home (root directory) you want to use. For complete details about SDK support, see *WebLogic Platform 8.1 Supported Configurations* at the following URL:

http://e-docs.bea.com/platform/supponfigs/configs81/81_over/overview.html

- Reliable Web services should not be used in an application deployed to a cluster. This is current known limitation in the 8.1 release. If your application includes a Web service with methods that use reliable messaging, you need to disable reliable messaging for those methods before you deploy. For more information, see “Java Web Service Annotations” in the *WebLogic Workshop Help* at the following URL:

Preparing the Application for Deployment

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/reference/tags/navJwsAnnotations.html>

- When you compile an EAR file using Build EAR, a `wlw-manifest.xml` file is produced and placed in the application's `META-INF` directory. This `wlw-manifest.xml` file lists the server resources that must be created on the production server for the application EAR to run successfully.

For more information about packaging applications for deployment, see the following topics:

- “Deploying an Application to a Production Server” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/guide/deployment/navDeployingApplications.html>

- “wlwBuild Command” in the *WebLogic Workshop Help* at the following URL:

<http://e-docs.bea.com/workshop/docs81/doc/en/workshop/reference/commands/cmdWlwBuild.html>

Deploying the Application

Once the target environment is set up and the application is prepared for deployment, the final step is to deploy the application to the target environment.

This section describes the following topics:

- [“About Deployment Units” on page 8-1](#)
- [“Overview of the Deployment Tools” on page 8-2](#)
- [“Deployment Considerations” on page 8-3](#)
- [“Steps to Deploy the Application” on page 8-7](#)
- [“Starting the Servers” on page 8-9](#)
- [“Example: How to Deploy a WebLogic Integration Application Using weblogic.Deployer and Ant” on page 8-11](#)
- [“Example: How to Deploy WebLogic Platform, WebLogic Portal, and WebLogic Integration Applications Using weblogic.Deployer and Ant” on page 8-13](#)

About Deployment Units

A *deployment unit* refers to a J2EE application (an Enterprise Application or Web Application) or a standalone J2EE module (an EJB or Resource Adapter) that has been organized according to the J2EE specification and can be deployed to WebLogic Server.

For each type of deployment unit, the J2EE specification defines both the required files and their location in the directory structure of the application or module. Deployment units may include

Java classes for EJBs and servlets, resource adapters, Web pages and supporting files, XML-formatted deployment descriptors, and even other modules.

J2EE does not specify *how* a deployment unit is deployed on the target server—only how standard applications and modules are organized. WebLogic Server supports deployments that are packaged either as archive files using the jar utility, or as exploded archive directories.

In certain circumstances, you may need to deploy individual modules within an application to different targets, for example, when deploying a WebLogic Platform application that combines multiple components of WebLogic Platform. Important considerations for targeting these modules and applications are provided in this section.

Overview of the Deployment Tools

The following table lists the deployment tools that are available with WebLogic Platform. You can use these tools to deploy, redeploy, undeploy, and distribute applications.

Table 8-1 WebLogic Platform Deployment Tools

This tool...	Enables you to...
WebLogic Server Administration Console	Perform basic deployment functions interactively using a browser. This method of deployment is convenient if you do not know the exact names of deployment units, target servers, or deployed applications. For more information about deploying applications using the WebLogic Server Administration Console, see Configuring and Managing WebLogic Server .
<code>weblogic.Deployer</code>	Provides a command-line based interface for performing both basic and advanced deployment tasks. Use <code>weblogic.Deployer</code> to automate deployment tasks using shell scripts or batch processes. For more information, see “ Deployment Tools Reference ” in <i>Deploying WebLogic Server Applications</i> .
<code>wldeploy</code>	Provides an Ant task version of the <code>weblogic.Deployer</code> utility. Use <code>wldeploy</code> to automate deployment tasks by including commands in an Ant <code>build.xml</code> file and running Ant to execute the commands. For more information, see “ Deployment Tools Reference ” in <i>Deploying WebLogic Server Applications</i> .

Deployment Considerations

The following sections provide considerations for deploying to a production environment, including:

- “Archive Type” on page 8-3
- “Application Targets” on page 8-3
- “Security Roles” on page 8-4
- “Staging Modes” on page 8-5
- “Load Order” on page 8-6
- “Deployment Descriptors” on page 8-6

Note: The Customer Support Web site provides general considerations and troubleshooting tips for deploying applications. For more information, see *Troubleshooting Deployment Issues* at:

http://support.bea.com/support_news/product_troubleshooting/Deployment_Pattern.html

Archive Type

You can store an application as a single archive EAR file or as an exploded archive directory. To review considerations, see “Deployment Files” in *Deploying WebLogic Server Applications*.

Application Targets

How you target the application depends on the application type:

- WebLogic Platform or WebLogic Portal application

Target the WebLogic Portal application or Portal modules of a WebLogic Platform application, to the Administration Server and the cluster. This enables data in the Administration Portal, the Datasync Web application, and the LDAP server to be synchronized and promoted correctly.

Deployment of a WebLogic Portal application or Portal modules of a WebLogic Platform application, is a two-step deployment process. For more information, see “Step 3: Deploy the Application” on page 8-8.

Note: WebLogic Portal does not support a split configuration architecture where EJBs and JSPs are split onto different servers in a cluster.

- WebLogic Integration application

Target the WebLogic Integration application to a *single* cluster in the domain. If no target cluster is specified, the target defaults to the first WebLogic Integration cluster or, in the absence of a cluster, the first Managed Server. You can modify this value by editing the `wli-config.properties` file to set the `weblogic.wli.WliClusterName` property value to the name of the cluster to which you want to target. For more information about the `wli-config.properties` file, see “[wli-config.properties Configuration File](#)” in *Deploying WebLogic Integration Solutions*.

When using a process control, the target process must be targeted to the same Managed Server as the client process. Otherwise, the dispatching table will not be updated and the client process will not have access to the information necessary to call the target process. Similarly, subscriber processes must be targeted to the same Managed Server as the publisher process.

Trading Partner Integration components must be deployed homogeneously to a cluster so that there is no single point of failure.

For considerations when deploying adapters and event generators, see the following sections in “[Understanding WebLogic Integration Clusters](#)” in *Deploying WebLogic Integration Solutions*:

- “Deploying Adapters”
- “Deploying Event Generators”

- WebLogic Workshop or WebLogic Server application

Target as required by the application. No specific restrictions apply.

For more information, see the following sections in *Deploying WebLogic Server Applications*:

- Deployment Targets in “[Overview of WebLogic Server Deployments](#)”
- “[Deploying Modules of an Enterprise Application to Different WebLogic Server Instances](#)” in *Deploying WebLogic Server Applications*.

Security Roles

The default Authentication provider delivers two built-in security roles, `Admin` and `Deployer`, to enable users to perform deployment tasks using the WebLogic Server Administration Console.

In addition, you may need to create a custom Authentication provider and define additional security roles to further secure the environment. BEA recommends that you create all roles

required by an application before deploying it. For more information, see [“Configuring Security” on page 5-1](#).

Staging Modes

The following table lists the staging modes that define how deployment units are made available to targeted servers. For more details about the staging modes and suggestions on when to use them, see [“Staging Modes”](#) in *Deploying WebLogic Server Applications*.

Table 8-2 Staging Modes

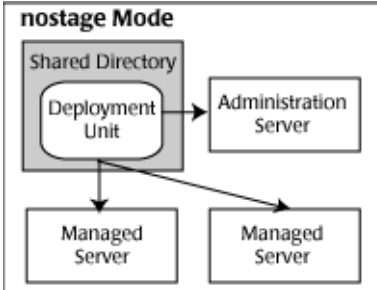
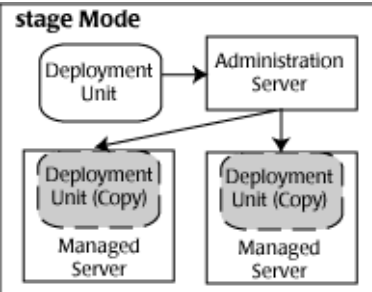
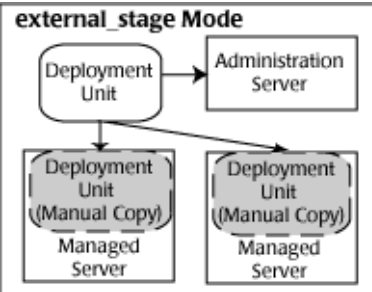
Staging Modes	Description
 <p>The diagram illustrates the 'nostage Mode'. It shows a 'Shared Directory' containing a 'Deployment Unit'. An arrow points from the 'Deployment Unit' to an 'Administration Server'. Two arrows point from the 'Shared Directory' to two separate 'Managed Server' boxes.</p>	<p>The deployment units are deployed using the same physical copy, which must be accessible by the Administration Server and target servers (for example, via a shared directory). The Administration Server does not copy the deployment unit files to the target servers, in this case. This mode is useful when deploying very large deployments to multiple targets, and for deployments that require dynamic updates. This is the default staging mode for the Administration Server.</p>

Table 8-2 Staging Modes (Continued)

Staging Modes	Description
 <p>The diagram for 'stage Mode' shows a 'Deployment Unit' box at the top left. An arrow points from it to an 'Administration Server' box at the top right. From the 'Administration Server', two arrows point down to two separate 'Managed Server' boxes. Each 'Managed Server' box contains a 'Deployment Unit (Copy)' box.</p>	<p>The Administration Server copies the deployment unit files to the staging directories of target servers and they are deployed using the local copy. This mode is useful when deploying small or moderate size applications, and prevents having a single point of failure if the original copy is not accessible. This is the default staging mode for Managed Servers.</p>
 <p>The diagram for 'external_stage Mode' shows a 'Deployment Unit' box at the top left. An arrow points from it to an 'Administration Server' box at the top right. Another arrow points from the 'Deployment Unit' box down to two separate 'Managed Server' boxes. Each 'Managed Server' box contains a 'Deployment Unit (Manual Copy)' box.</p>	<p>In external_stage mode you must copy the deployment units manually to the correct staging directories before deployment. Use this staging mode for deployments where you want to manually control the distribution of deployment files to target servers. This mode prevents deployment unit information from being dynamically updated. In this case, the Administration Server accesses the original (master) deployment unit for validation.</p>

Load Order

The deployment order is determined by the Load Order attribute. By default, each deployment unit is configured with a Load Order value of 100. Deployment units with the same Load Order value are deployed in alphabetical order using the deployment name.

You can change the load order in the WebLogic Server Administration Console or using the `ApplicationMBean`. For more information, and for a description of configuring the application module and startup class order, see [“Deployment Order”](#) in *Deploying WebLogic Server Applications*.

Deployment Descriptors

To adjust the run-time deployment configuration without modifying the contents of the application archive, you can use alternate deployment descriptors that are staged externally to the

application. You can specify an alternate deployment descriptor file to be used in place of the standard J2EE (`application.xml`) or WebLogic Server (`weblogic-application.xml`) deployment descriptor.

For more information, see [“Deploying Enterprise Applications With Alternate Deployment Descriptors”](#) in *Deploying WebLogic Server Applications*.

Steps to Deploy the Application

The following sections describe the steps that are required to deploy applications to the target environment, including:

- [Step 1: Start the Servers](#)
- [Step 2: Upload the Application to the Administration Server \(Optional\)](#)
- [Step 3: Deploy the Application](#)
- [Step 4: Deploy Event Generators](#)

Note: Before deploying to the target environment, be sure that you have prepared the target environment and the application, as described in the following sections:

- [“Creating and Configuring the WebLogic Domain” on page 3-1](#)
- [“Configuring the Production Database” on page 4-1](#)
- [“Configuring Security” on page 5-1](#)
- [“Using Load Balancers and Web Proxy Servers” on page 6-1](#)
- [“Preparing the Application for Deployment” on page 7-1](#)

Step 1: Start the Servers

Start the Administration Server and Managed Servers, if they are not already running, as described in [“Starting the Servers” on page 8-9](#).

Step 2: Upload the Application to the Administration Server (Optional)

In order to deploy an application to servers in a domain, the files must be accessible to the Administration Server for the domain. Specifically, they must reside on the Administration Server machine or be accessible via a network mounted directory. For more information, see

“Uploading Deployment Files to the Administration Server” in [“Performing Common Deployment Tasks”](#) in *Deploying WebLogic Server Applications*.

Step 3: Deploy the Application

The procedure that you use to deploy the application depends on the application type. WebLogic Platform and WebLogic Portal applications require a special deployment sequence.

The following sections describe the procedures for deploying an application, based on the application type:

- [Deploying a WebLogic Integration, WebLogic Server, or WebLogic Workshop Application](#)
- [Deploying a WebLogic Platform or WebLogic Portal Application](#)

Deploying a WebLogic Integration, WebLogic Server, or WebLogic Workshop Application

Deploy the WebLogic Integration, WebLogic Server, or WebLogic Workshop application to the cluster using one of the deployment tools described in [“Overview of the Deployment Tools”](#) on page 8-2. For targeting and other deployment considerations, see [“Deployment Considerations”](#) on page 8-3.

Deploying a WebLogic Platform or WebLogic Portal Application

Deployment of a WebLogic Portal application or Portal modules of a WebLogic Platform application, is a two-step deployment process. Initially, the WebLogic Portal application or Portal modules must be deployed to a single server and then subsequently deployed to the cluster.

To deploy a WebLogic Platform or WebLogic Portal Application:

1. Deploy the WebLogic Portal application or Portal modules of a WebLogic Platform application initially to the Administration Server only.
2. Deploy the WebLogic Portal or WebLogic Platform application to the cluster.

For targeting and other deployment considerations, see [“Deployment Considerations”](#) on page 8-3.

Note: If your cluster is not already running, you can make use of *deferred deployment*. To initiate deferred deployment, deploy the WebLogic Portal application or Portal modules of a WebLogic Platform application to the Administration Server and the cluster; deploy all other applications and modules to the cluster only. Then, start the Managed Servers, as described in [“Starting the Managed Servers”](#) on page 8-11. At this point, the

applications that were deployed to the cluster, are deployed automatically. One advantage of deferred deployment is that the WebLogic Portal application or modules only need to be deployed once.

Step 4: Deploy Event Generators

If your application uses event generators, you must deploy them using the WebLogic Integration Administration Console. For information about deploying event generators, see [“Creating and Deploying Event Generators”](#) in “Event Generators” in *Managing WebLogic Integration Solutions*.

Starting the Servers

You can start the Administration Server and Managed Servers, as described in the following sections:

- [Before You Start the Servers](#)
- [Starting the Administration Server](#)
- [Starting the Managed Servers](#)

Before You Start the Servers

Before you start the servers, perform the following tasks:

1. Set up the Managed Server directories, as described in [“Setting Up the Managed Server Directories”](#) on page 3-15.
2. If you will be using the Node Manager:
 - a. Configure the Node Manager, as described in [“Configuring Node Manager”](#) on page 3-21.
 - b. Start the Node Manager on each machine in one of the following ways:
 - Automatically, as a WebLogic service—The WebLogic Server installation process automatically installs Node Manager as a service, so that it starts up automatically when the system boots.
 - From a command-line or script—Although running Node Manager as an operating system service is recommended, you can also start Node Manager manually at the command prompt or with a script.

Sample start scripts for Node Manager are installed in the `WL_HOME\server\bin` directory, where `WL_HOME` is the top-level installation directory for WebLogic Server. Use `startNodeManager.cmd` on Windows systems and `startNodeManager.sh` on UNIX systems.

For more information about starting and stopping Node Manager, see [“Configuring, Stopping, and Starting Node Manager”](#) in *Configuring and Managing WebLogic Server*.

3. For a WebLogic Platform or WebLogic Portal application, before starting the servers, consider increasing the default memory size to 512 MB, at a minimum. To do so, update all references to the `MEM_ARGS` variable in the `setDomainEnv` script in the domain directory, as follows:

- Windows: `set MEM_ARGS=Xms512m -Xm512m`
- UNIX: `MEM_ARGS="-Xms512m -Xm512m"`

If you are using Node Manager to administer Managed Servers, set this variable using the `Arguments ServerStart` attribute, as described in [“Configuring the Managed Server Start Attributes”](#) on page 3-16.

Starting the Administration Server

To start the Administration Server:

1. Open a shell (command prompt) on the computer on which you created the domain.
2. Change to the directory in which you created the domain.

For example: `BEA_HOME\user_projects\domains\domain-name`, where `BEA_HOME` specifies the directory where you installed WebLogic Platform 8.1, and `domain-name` specifies the name of the domain.

3. Run the available start-up script.

For example:

Windows: `startWebLogic.cmd`

UNIX: `startWebLogic.sh`

Note: If you used a Configuration Wizard template that is provided by WebLogic Platform to create the domain, the domain directory includes a start script, typically named `startWebLogic`.

For more information about starting the Administration Server, see [“Starting Administration Servers”](#) in *WebLogic Server Administration Console Online Help*.

Starting the Managed Servers

Start the Managed Server in one of the following ways:

- If you are using Node Manager, start the Managed Server as described in [“Overview of Node Manager”](#) in *Configuring and Managing WebLogic Server*.
- If you are not using Node Manager, then you must have created a domain directory on the Managed Server. Execute the `startManagedWebLogic` script provided in the domain directory.

Example: How to Deploy a WebLogic Integration Application Using `weblogic.Deployer` and Ant

The following example provides a code excerpt from an Ant build file and demonstrates how to use `weblogic.Deployer` to deploy a WebLogic Integration application in an automated way to a single-cluster domain, as shown in [“Single-Cluster Platform Domain Example”](#) on page 2-9.

The following sections step you through the process of deploying an application to a target domain.

- [“Define the Properties in the Ant Script”](#) on page 8-11
- [“Define the Main Ant Target”](#) on page 8-12
- [“Define the Deploy Ant Target”](#) on page 8-12
- [“Deploy the WebLogic Integration Application”](#) on page 8-12

Define the Properties in the Ant Script

To facilitate automation and reuse of the scripts in different target environments, the code excerpt references *properties*, such as `${cluster.name}`, that are resolved in a properties file imported to the Ant script. For example, the following properties are defined in a separate properties file, `myprops.properties`:

```
deploy.dir=deploy
cluster.name=platformcluster
admin.addr=myhost
admin.port=9301
admin.username=username
admin.password=password
```

The file is referenced in the Ant build file as follows:

Deploying the Application

```
<property file="myprops.properties"/>
```

Define the Main Ant Target

The following defines the main Ant target for the WebLogic Integration application. The target value is shown in **bold**. The `deploy-app` target is described below.

```
<target name="deploy-IntApp">
  <antcall target="deploy-app">
    <param name="app.name" value="IntApp" />
    <param name="app.src" location="${deploy.dir}/IntApp.ear" />
    <param name="app.target" value="_${cluster.name}" />
    <param name="admin.url" value="http://${admin.addr}:${admin.port}" />
  </antcall>
</target>
```

Define the Deploy Ant Target

The `deploy-app` target, referenced in the Ant target above, uses the `weblogic.Deployer` command to deploy the applications, as follows:

```
<target name="deploy-app">
  <java classname="weblogic.Deployer"
    fork="true" failonerror="true">
    <arg value="-adminurl"/><arg value="${admin.url}"/>
    <arg value="-username"/><arg value="${admin.username}"/>
    <arg value="-password"/><arg value="${admin.password}"/>
    <arg value="-name"/><arg value="${app.name}"/>
    <arg value="-source"/><arg value="${app.src}"/>
    <arg value="-targets"/><arg value="_${app.target}"/>
    <arg value="-deploy"/>
  </java>
```

Deploy the WebLogic Integration Application

To deploy the WebLogic Integration application to a single domain, as depicted in “[Single-Cluster Platform Domain Example](#)” on page 2-9:

1. Start the Administration Server and Managed Servers, as described in “[Steps to Deploy the Application](#)” on page 8-7.
2. Deploy the WebLogic Integration application using the main Ant target, as follows:

```
ant deploy-IntApp
```

Example: How to Deploy WebLogic Platform, WebLogic Portal, and WebLogic Integration Applications Using `weblogic.Deployer` and Ant

The following examples provide code excerpts from an Ant build file and demonstrates how to use `weblogic.Deployer` to deploy applications in an automated way to a single-cluster domain, as shown in [“Single-Cluster Platform Domain Example” on page 2-9](#).

Specifically, this example deploys the following three applications to a production environment:

- `PlatApp`—WebLogic Platform application. For a definition, see [“About WebLogic Platform Applications” on page 1-1](#).
- `PortApp`—WebLogic Portal application.
- `IntApp`—WebLogic Integration application.

The following sections step you through the process of deploying an application to a target domain.

- [“Define the Properties in the Ant Script” on page 8-13](#)
- [“Define the Ant Targets” on page 8-14](#)
- [“Define the Deploy Ant Targets” on page 8-15](#)
- [“Deploy the WebLogic Platform, WebLogic Portal, and WebLogic Integration Applications” on page 8-15](#)

Define the Properties in the Ant Script

To facilitate automation and reuse of the scripts in different target environments, the code excerpt references *properties*, such as `${cluster.name}`, that are resolved in a properties file imported to the Ant script. For example, the following properties are defined in a separate properties file, `myprops.properties`:

```
cluster.name=platformcluster
admin.name=cgServer
```

The file is referenced in the Ant build file as follows:

```
<property file="myprops.properties"/>
```

Define the Ant Targets

As described in “[Step 3: Deploy the Application](#)” on page 8-8, deploying a WebLogic Platform or WebLogic Portal application is a two-step deployment process (unless you use are using deferred deployment):

1. Deploy the WebLogic Portal application and Portal modules of a WebLogic Platform application initially to the Administration Server only.
2. Deploy the WebLogic Platform or WebLogic Portal application to the cluster.

For example, the following code excerpt defines the “first” Ant targets, to accomplish step 1 above. Target values are shown in **bold**. Note that `deploy-PlatApp-first` uses module-level targeting to target the WebLogic Portal modules of the WebLogic Platform application to the Administration Server only and WebLogic Integration modules to the cluster.

Note: The `deploy-app` target referenced in the following Ant targets is defined in “[Define the Deploy Ant Target](#)” on page 8-12.

You do not need to define a “first” Ant target for the WebLogic Integration application because it does not require a two-step deployment procedure.

```
<target name="deploy-PlatApp-first">
  <antcall target="deploy-PlatApp">
    <param name="app.target"
value="p13n_ejb.jar@${admin.name}, PortIntAppDatasync@${admin.name}, \
.workshop/PortIntAppWeb/EJB/GenericStateless@${cluster.name}, \
.workshop/PortIntAppWeb/EJB/Allocate_1trtqtoxcz4uv@${cluster.name}, \
.workshop/PortIntAppWeb/EJB/ProjectBeans@${cluster.name}, PortIntAppWeb@${clust
er.name}"
    />
  </antcall>
</target>

<target name="deploy-PortApp-first">
  <antcall target="deploy-PortApp">
<param name="app.target" value="${admin.name}"
  </antcall>
</target>
```

The following defines the main Ant targets to accomplish step 2 above. Target values are shown in **bold**. Note that the WebLogic Portal and WebLogic Platform applications are targeted to the cluster, since they will already have been targeted to the Administration Server via the “first” Ant targets described above.

Note: The main Ant target for the WebLogic Integration application is defined in “[Define the Main Ant Target](#)” on page 8-12.

```
<target name="deploy-PlatApp">
  <antcall target="deploy-app">
    <param name="app.name" value="PlatApp" />
    <param name="app.src" location="${deploy.dir}/PortIntApp.ear" />
    <param name="app.target" value="${cluster.name}" />
    <param name="admin.url" value="http://${admin.addr}:${admin.port}" />
  </antcall>
</target>

<target name="deploy-PortApp">
  <antcall target="deploy-app">
    <param name="app.name" value="PortApp" />
    <param name="app.src" location="${deploy.dir}/PortApp.ear" />
    <param name="app.target" value="${cluster.name}" />
    <param name="admin.url" value="http://${admin.addr}:${admin.port}" />
  </antcall>
</target>
```

Define the Deploy Ant Targets

Two deployment Ant targets are defined below: 1) One to execute the “first” Ant targets and deploy the WebLogic Portal application and Portal modules of the WebLogic Platform application to the Administration Server only, and 2) The second to deploy the WebLogic Portal, WebLogic Platform, and WebLogic Integration applications to the cluster.

- First deployment Ant target:

```
<target name="deploy-apps-first"
  depends="
    deploy-PlatApp-first,
    deploy-PortApp-first
  />
```

- Main deployment Ant target:

```
<target name="deploy-apps"
  depends="
    deploy-PlatApp
    deploy-PortApp
    deploy-IntApp
  />
```

Deploy the WebLogic Platform, WebLogic Portal, and WebLogic Integration Applications

To deploy the applications to a single domain, as depicted in “[Single-Cluster Platform Domain Example](#)” on page 2-9:

Deploying the Application

1. Start the Administration Server and Managed Servers, as described in [“Steps to Deploy the Application” on page 8-7](#).
2. Deploy the WebLogic Portal and WebLogic Platform applications initially using the “first” Ant target, as follows:

```
ant deploy-apps-first
```

3. Deploy the applications using the main Ant target, as follows:

```
ant deploy-apps
```

Deployment Targeting Reference

This appendix provides a comprehensive reference to the targets required for system-level applications and services to support a WebLogic Platform application in a production deployment environment. This information may be helpful for troubleshooting purposes. Topics include:

- [Characteristics of a Production Deployment Domain](#)
- [Distribution of Production Domain Resources](#)
- [Deployment Targeting Reference - Single Cluster WebLogic Platform Domain](#)
- [Default Domain Resource Reference - By Product Component](#)

Characteristics of a Production Deployment Domain

The production deployment domain described in this section is based on the following assumptions and characteristics:

- It is created using the Configuration Wizard or WLST Offline, both of which have underlying autoconfiguration capabilities to simplify domain creation. For more information about autoconfiguration, see [“Autoconfiguration Using the Configuration Wizard and WLST Offline” on page 3-12](#).
- It is created from the Basic WebLogic Platform Domain template.
- It has multiple Managed Servers.
- It has a single cluster to which all the Managed Servers are assigned.

- All Managed Servers in the cluster are autoconfigured as migratable.
- Other resources and services, such as JMS servers and distributed JMS queues, are autoconfigured.

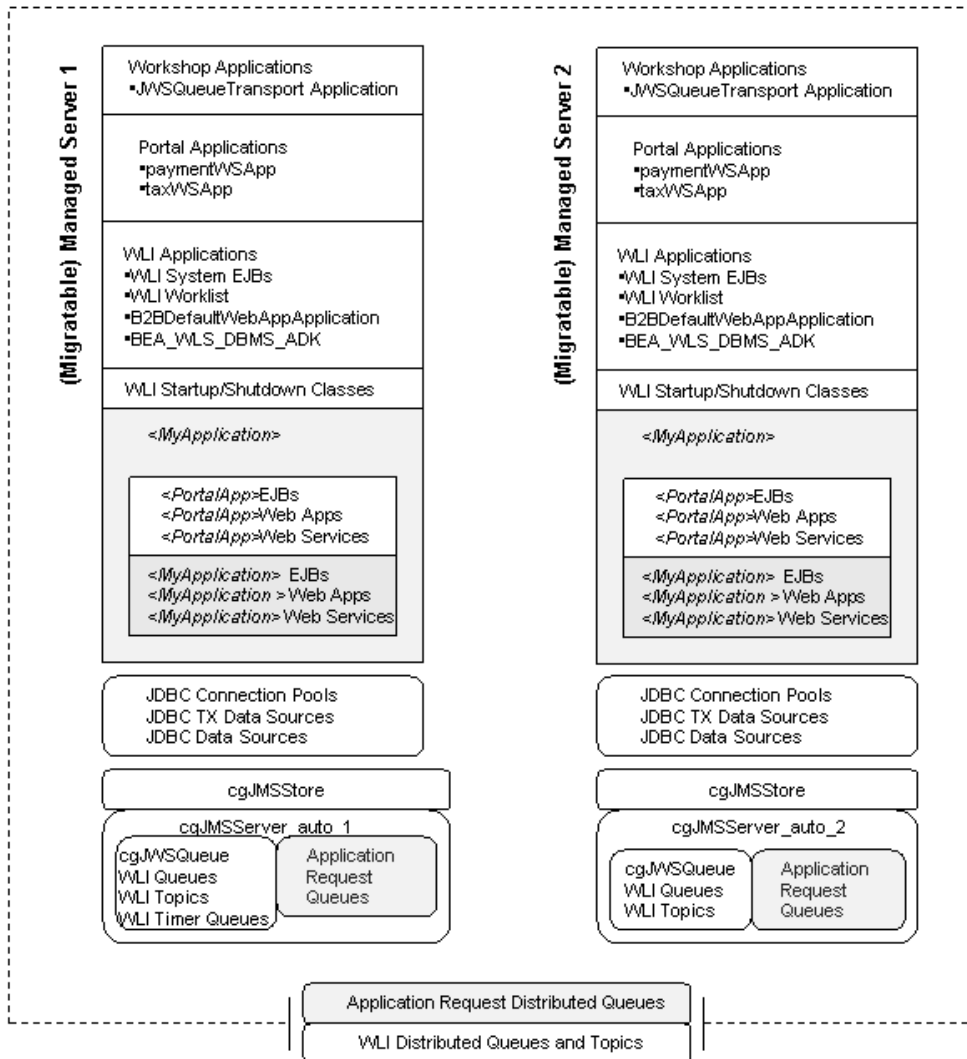
Distribution of Production Domain Resources

The following illustrations show the basic distribution of system-level resources, services, and applications in a single-cluster WebLogic Platform domain. Also illustrated is the distribution of resources for a Platform application. A Platform application is defined to be a composite of WebLogic Portal and WebLogic Integration components.

- [Figure A-1](#) shows the distribution of resources across the Managed Servers and the cluster.
- [Figure A-2](#) shows the resources targeted to the domain's Administration Server.

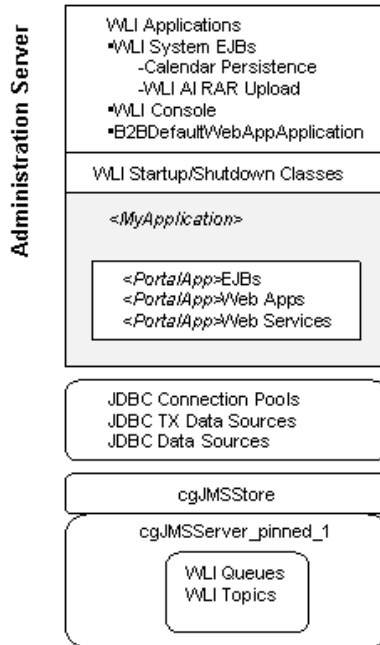
Note: Shaded areas in the illustrations represent your application and the resources that may be required to support your application in the domain.

Figure A-1 Resource Distribution Across Managed Servers in a Single Cluster WebLogic Platform Domain



Note: The main difference between the resources depicted on the Managed Servers in this illustration is that only the JMS Server, cgJMSServer_auto_1, which is associated with (Migratable) Managed Server 1, includes WLI Timer Queues. These WLI Timer Queues are targeted to the JMS Server on only one Managed Server in a domain.

Figure A-2 Resources on the Administration Server in a Single Cluster WebLogic Platform Domain



Deployment Targeting Reference - Single Cluster WebLogic Platform Domain

The following table provides a detailed reference to the resources, services, and application targeting required for a single-cluster Platform domain. Note the following:

- *<MyApplication>* represents the name of an application EAR file being deployed.
- *<MyWebApp>* represents the name of an individual Web application developed in WebLogic Workshop.
- *<MyPortalApp>* represents the name of an individual portal application developed in WebLogic Workshop.
- The “#” represents an integer in the name of a resource, such as a Managed Server, JMS server, or JMS queue, that is autoconfigured by the Configuration Wizard or WLST

Offline during clustered domain setup. For example, the JMS server provided in the Basic WebLogic Platform Domain template is `cgJMSServer`. As such, the name of the JMS server autoconfigured for the first Managed Server is `cgJMSServer_auto_1`; the name of the JMS server autoconfigured for the second Managed Server is `cgJMSServer_auto_2`, and so on. Note also, for resources that must be pinned to a specific server, such as the Administration Server, the autoconfiguration process specifies “pinned” in the name of the resource; for example, `cgJMSServer_pinned_1` is the name of the JMS Server autoconfigured and pinned to the Administration Server in a cluster.

- A check mark indicates the default target for the resource, service, or application.

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Adminis- tration Server	One Managed Server Only	Each Managed Server	Cluster
Applications					
	<code><MyApplication>*</code>	✓*			✓
	*Target to both the Administration Server and the cluster when Portal components are included in the application. Target only to the cluster when Portal components are not included in the application. See <code><MyPortalApp></code> for Portal-only application targeting				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
<i><MyPortalApp></i>					
	EJB Components	✓			✓
	<ul style="list-style-type: none"> • commerce.jar (if commerce services installed) • content.jar • content_repo.jar • netuix.jar • p13n_ejb.jar • pipeline.jar (if pipeline services installed) • prefs.jar • wps.jar 				
	Web App Components	✓			✓
	<ul style="list-style-type: none"> • <i><MyPortalApp></i> Admin • <i><MyPortalApp></i> Datasync • <i><MyPortalApp></i> 				
	Web Service Components	✓			✓
	<ul style="list-style-type: none"> • <i><MyPortalApp></i> Tool • <i><MyPortalApp></i> WPSTool 				
<i>paymentWSApp</i>					
	EJB Components				✓
	<ul style="list-style-type: none"> • payment 				
	Web Service Components				✓
	<ul style="list-style-type: none"> • payws 				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Adminis- tration Server	One Managed Server Only	Each Managed Server	Cluster
	taxWSApp				
	EJB Components				✓
	• tax				
	Web Service Components				✓
	• taxws				
	WLI-AI Design-time (Present in the domain, but application components are not targeted by default)				
	EJB Components				
	• WLI-AI Manager EJBs				
	Web App Components				
	• wlai				
	JWSQueueTransport				
	EJB Components				✓
	• QueueTransportEJB				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Adminis- tration Server	One Managed Server Only	Each Managed Server	Cluster
WLI System EJBs					
	EJB Components				✓
	<ul style="list-style-type: none"> • WLI Admin • WLI Admin Helper • WLI AI Message Processors • WLI ebXML • WLI Message Tracking • WLI Process Proxy Dispatcher • WLI Process Tracking • WLI RosettaNet • WLI Sync2Async Response Listener • WLI Worklist Persistence • WLI Worklist Selection 				
	<ul style="list-style-type: none"> • WLI Calendar Persistence 	✓			✓
	<ul style="list-style-type: none"> • WLI AI RAR Upload 	✓			
	Web App Components				✓
	<ul style="list-style-type: none"> • WLI-B2B HTTP Transport • WLI Sync2Async Transport Servlet 				
WLI Console					
	Web App Components		✓		
	<ul style="list-style-type: none"> • wliconsole 				
WLI Worklist					

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
	EJB Components				✓
	<ul style="list-style-type: none"> .workshop/worklist/EJB/GenericStateless .workshop/worklist/EJB/ProjectBeans 				
	Web App Components				✓
	<ul style="list-style-type: none"> worklist 				
	B2BDefaultWebAppApplication				
	Web App Components	✓			✓
	<ul style="list-style-type: none"> B2BDefaultWebApp 				
	BEA_WLS_DBMS_ADK				
	Web App Components				✓
	<ul style="list-style-type: none"> BEA_WLS_DBMS_ADK_Web 				
	Connector Components				✓
	<ul style="list-style-type: none"> BEA_WLS_DBMS_ADK BEA_WLS_DBMS_ADK_LOCALTX 				
	Startup and Shutdown Classes				
	<ul style="list-style-type: none"> WLI Post-Activation Startup Class WLI Shutdown Class WLI Startup Class 	✓			✓
	JMS Servers				
	cgJMSServer_pinned_1	✓			
	cgJMSServer_auto_1		✓		
	Note: Certain JMS queues are associated with only one JMS Server (that is, cgJMSServer_auto_1) that is targeted to the first Managed Server.				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
	cgJMSServer_auto_#			✓	
JMS Connection Factories					
	cgQueue	✓			✓
	WLI-B2B System Topic Factory	✓			
	wli.internal.egrdbms.QueueConnectionFactory				✓
JMS Queues					
	<i>JMS Queues—for each managed server in the cluster for each <MyWebApp> created in WebLogic Workshop where Web services are configured to support asynchronous requests.</i>			✓	
	<ul style="list-style-type: none"> • <MyWebApp>.queue.AsyncDispatcher<1...n>_# • <MyWebApp>.queue.AsyncDispatcher_error<1...n>_# 				
	JMS Queues—for cgJMSServer_pinned_1	✓			
	<ul style="list-style-type: none"> • wli.internal.configfile.request.queue 				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
	JMS Queues—for each managed server in the cluster			✓	
	<ul style="list-style-type: none"> • cgJWSQueue_auto_# • wli.b2b.failedmessage.queue_auto_# • wli.b2b.mt.event.stream_auto_# • wli.b2b.mt.event.stream_error_auto_# • wli.internal.ai.async.request_auto_# • wli.internal.ai.async.response_auto_# • wli.internal.ai.event_suspend_auto_# • wli.internal.b2b.ebxmlencoder.queue_auto_# • wli.internal.b2b.rosettanetencoder.queue_auto_# • wli.internal.egmq.queue_auto_# • wli.internal.egrdbms.queue_auto_# • wli.internal.instance.info.buffer_auto_# • wli.internal.instance.info.buffer_error_auto_# • wli.internal.msgtracking.queue_auto_# • wli.internal.sync2Async.soapResponse_auto_# • wli.internal.tracking.buffer_auto_# • wli.internal.tracking.buffer_error_auto_# • wli.internal.worklist.timer.queue_auto_# • wli.process.event.stream_auto_# • wli.process.event.stream_error_auto_# • wli.sample.egjms.queue_auto_# 				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Adminis- tration Server	One Managed Server Only	Each Managed Server	Cluster
	JMS Queues—Unique to the JMS server on one Managed Server in the cluster (e.g., cgJMSServer_auto_1 on new_Server_1)		✓		
	<ul style="list-style-type: none"> • wli.internal.egfile.queue • wli.internal.egmail.queue • wli.internal.egtimer.queue • wli.internal.scheduling.queue • wli.internal.scheduling.queue_error • wli.internal.SQLStore.cleanup.documents 				
JMS Topics					
	JMS Topics—for cgJMSServer_pinned_1	✓			
	<ul style="list-style-type: none"> • wli.internal.b2bevents.topic • wli.internal.configfile.update.topic 				
	<ul style="list-style-type: none"> • wli.internal.ai.event_auto_# 			✓	
JMS Distributed Queues					
	<i>JMS Distributed Queues—for each <MyWebApp> created in WebLogic Workshop where Web services are configured to support asynchronous request.</i>				✓
	<ul style="list-style-type: none"> • <i>dist_<MyWebApp>.queue.AsyncDispatcher<1...n></i> • <i>dist_<MyWebApp>.queue.AsyncDispatcher_error<1...n></i> 				

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
	<ul style="list-style-type: none"> • dist_cgJWSQueue_auto • dist_wli.b2b.failedmessage.queue_auto • dist_wli.b2b.mt.event.stream_auto • dist_wli.b2b.mt.event.stream_error_auto • dist_wli.internal.ai.async.request_auto • dist_wli.internal.ai.async.response_auto • dist_wli.internal.ai.event_suspend_auto • dist_wli.internal.b2b.ebxmlencoder.queue_auto • dist_wli.internal.b2b.rosettanetencoder.queue_auto • dist_wli.internal.egmq.queue_auto • dist_wli.internal.egrdbms.queue_auto • dist_wli.internal.instance.info.buffer_auto • dist_wli.internal.instance.info.buffer_error_auto • dist_wli.internal.msgtracking.queue_auto • dist_wli.internal.sync2Async.soapResponse_auto • dist_wli.internal.tracking.buffer_auto • dist_wli.internal.tracking.buffer_error_auto • dist_wli.internal.worklist.timer.queue_auto • dist_wli.process.event.stream_auto • dist_wli.process.event.stream_error_auto • dist_wli.sample.egjms.queue_auto 				✓
JMS Distributed Topics					
	dist_wli.internal.ai.event_auto				✓
JMS JDBC Data Store					

Table A-1 Default Targets for Resources, Services, and Applications in a Clustered Platform Domain

Item	Name	Administration Server	One Managed Server Only	Each Managed Server	Cluster
	cgJMSSStore_auto_#	✓			
	Note: One JMS JDBC Data Store is assigned to the JMS Server (that is, cgJMSServer_pinned_1) that is targeted to the Administration Server during autoconfiguration.				
	cgJMSSStore_auto_#			✓	
JDBC Connection Pools					
	<ul style="list-style-type: none"> • bpmArchPool • cgJMSPool-nonXA • cgPool • portalPool 	✓			✓
JDBC Data Sources					
	<ul style="list-style-type: none"> • p13n_trackingDataSource • p13nDataSource 	✓			✓
JDBC Tx Data Sources					
	<ul style="list-style-type: none"> • bpmArchDataSource • cgDataSource • cgDataSource-nonXA • portalFrameworkPool 	✓			✓

Default Domain Resource Reference - By Product Component

The following table provides a reference to the default resources, services, and applications provided in the Basic WebLogic Platform Domain template and their relationships to the platform’s constituent product components, such as WebLogic Portal and WebLogic Integration. The table also includes references to autoconfigured resources that result from using the Configuration Wizard or WLST Offline to create a clustered domain.

Use this reference to identify resources that you may need to configure or target manually when creating a product-specific cluster in a multi-cluster domain. For example, in a multi-cluster platform domain, where you may want one WebLogic Integration cluster and a separate WebLogic Portal cluster, you may need to retarget resources for each cluster to create separate product-specific clusters. If you should need to configure other resources manually, the following table can provide a guide as to which resources need to be configured for product-specific components. For an example that shows how to configure a multi-cluster platform domain, see [“Example: How to Configure a Multi-Cluster Platform Domain Using WLST Offline” on page 3-30.](#)

Note: In the following table, the “#” represents an integer in the name of a resource, such as a Managed Server, JMS server, or JMS queue, that is autoconfigured by the Configuration Wizard or WLST Offline during clustered domain setup. For example, the JMS server provided in the Basic WebLogic Platform Domain template is `cgJMSServer`. As such, the name of the JMS server autoconfigured for the first Managed Server is `cgJMSServer_auto_1`; the name of the JMS server autoconfigured for the second Managed Server is `cgJMSServer_auto_2`, and so on. Note also, for resources that must be pinned to a specific server, such as the Administration Server, the autoconfiguration process specifies “pinned” in the name of the resource; for example, `cgJMSServer_pinned_1` is the name of the JMS Server autoconfigured and pinned to the Administration Server in a cluster.

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
Applications					
	paymentWSApp				
	EJB Components		✓		✓
	• payment				
	Web Service Components		✓		✓
	• payws				

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
taxWSApp					
	EJB Components		✓		✓
	• tax				
	Web Service Components		✓		✓
	• taxws				
WLI-AI Design-time					
	EJB Components			✓	✓
	• WLI-AI Manager EJBs				
	Web App Components			✓	✓
	• wlai				
JWSQueueTransport					
	EJB Components	✓	✓	✓	✓
	• QueueTransportEJB				

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
WLI System EJBs					
	EJB Components			✓	✓
	<ul style="list-style-type: none"> • WLI Admin • WLI Admin Helper • WLI Calendar Persistence • WLI Process Tracking • WLI Worklist Persistence • WLI Worklist Selection • WLI Process Proxy Dispatcher • WLI Sync2Async Response Listener • WLI AI Message Processors • WLI RosettaNet • WLI ebXML • WLI Message Tracking • WLI AI RAR Upload 				
	Web App Components			✓	✓
	<ul style="list-style-type: none"> • WLI-B2B HTTP Transport • WLI Sync2Async Transport Servlet 				
WLI Console					
	Web App Components			✓	✓
	<ul style="list-style-type: none"> • wliconsole 				
WLI Worklist					
	EJB Components			✓	✓
	<ul style="list-style-type: none"> • .workshop/worklist/EJB/ProjectBeans • .workshop/worklist/EJB/GenericStateless 				
	Web App Components			✓	✓
	<ul style="list-style-type: none"> • worklist 				

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
B2BDefaultWebAppApplication					
	Web App Components			✓	✓
	• B2BDefaultWebApp				
BEA_WLS_DBMS_ADK					
	Web App Components			✓	✓
	• BEA_WLS_DBMS_ADK_Web				
	Connector Components			✓	✓
	• BEA_WLS_DBMS_ADK				
	• BEA_WLS_DBMS_ADK_LOCALTX				
Startup and Shutdown Classes					
	• WLI Post-Activation Startup Class			✓	✓
	• WLI Shutdown Class				
	• WLI Startup Class				
JMS Servers					
	cgJMSServer_pinned_1 (Administration Server)			✓	✓
	cgJMSServer_auto_# (On Managed Servers)	✓	✓	✓	✓
	Note: Certain JMS queues are associated with only one JMS Server (that is, cgJMSServer_auto_1) that is targeted to the first Managed Server.				
JMS Connection Factories					
	cgQueue	✓	✓	✓	✓
	WLI-B2B System Topic Factory			✓	✓
	wli.internal.egrdbms.QueueConnectionFactory			✓	✓

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
JMS Queues					
	cgJWSQueue_auto_#	✓	✓	✓	✓
	<ul style="list-style-type: none"> • wli.b2b.failedmessage.queue_auto_# • wli.b2b.mt.event.stream_auto_# • wli.b2b.mt.event.stream_error_auto_# • wli.internal.ai.async.request_auto_# • wli.internal.ai.async.response_auto_# • wli.internal.ai.event_suspend_auto_# • wli.internal.b2b.ebxmlencoder.queue_auto_# • wli.internal.b2b.rosettanetencoder.queue_auto_# • wli.internal.configfile.request.queue_auto_# • wli.internal.egfile.queue_auto_# • wli.internal.egmail.queue_auto_# • wli.internal.egmq.queue_auto_# • wli.internal.egrdbms.queue_auto_# • wli.internal.egtimer.queue_auto_# • wli.internal.instance.info.buffer_auto_# • wli.internal.instance.info.buffer_error_auto_# • wli.internal.msgtracking.queue_auto_# • wli.internal.scheduling.queue_auto_# • wli.internal.scheduling.queue_error_auto_# • wli.internal.SQLStore.cleanup.documents_auto_# • wli.internal.sync2Async.soapResponse_auto_# • wli.internal.tracking.buffer_auto_# • wli.internal.tracking.buffer_error_auto_# • wli.internal.worklist.timer.queue_auto_# • wli.process.event.stream_auto_# • wli.process.event.stream_error_auto_# • wli.sample.egjms.queue_auto_# 			✓	✓

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
JMS Topics					
	<ul style="list-style-type: none"> wli.internal.ai.event_auto_# wli.internal.b2bevents.topic wli.internal.configfile.update.topic 			✓	✓
JMS Distributed Queues					
	<ul style="list-style-type: none"> dist_cgJWSQueue_auto 	✓	✓	✓	✓
	<ul style="list-style-type: none"> dist_wli.b2b.failedmessage.queue_auto dist_wli.b2b.mt.event.stream_auto dist_wli.b2b.mt.event.stream_error_auto dist_wli.internal.ai.async.request_auto dist_wli.internal.ai.async.response_auto dist_wli.internal.ai.event_suspend_auto dist_wli.internal.b2b.ebxmlencoder.queue_auto dist_wli.internal.b2b.rosettanetencoder.queue_auto dist_wli.internal.egmq.queue_auto dist_wli.internal.egrdbms.queue_auto dist_wli.internal.instance.info.buffer_auto dist_wli.internal.instance.info.buffer_error_auto dist_wli.internal.msgtracking.queue_auto dist_wli.internal.sync2Async.soapResponse_auto dist_wli.internal.tracking.buffer_auto dist_wli.internal.tracking.buffer_error_auto dist_wli.internal.worklist.timer.queue_auto dist_wli.process.event.stream_auto dist_wli.process.event.stream_error_auto dist_wli.sample.egjms.queue_auto 			✓	✓
JMS Distributed Topics					

Table A-2 Default Domain Resources, Services, and Applications by Product Component

Item	Name	WebLogic Workshop	WebLogic Portal	WebLogic Integration	WebLogic Platform
	dist_wli.internal.ai.event_auto			✓	✓
JMS JDBC Data Store					
	cgJMSSStore_auto_#	✓	✓	✓	✓
	Note: One JMS JDBC Data Store is assigned to the JMS Server (that is, cgJMSServer_pinned_1) that is targeted to the Administration Server during autoconfiguration.				
JDBC Connection Pools					
	bpmArchPool			✓	✓
	<ul style="list-style-type: none"> • cgJMSPool-nonXA • cgPool 	✓	✓	✓	✓
	portalPool		✓		✓
JDBC Data Sources					
	<ul style="list-style-type: none"> • p13n_trackingDataSource • p13nDataSource 		✓		✓
JDBC Tx Data Sources					
	bpmArchDataSource			✓	✓
	<ul style="list-style-type: none"> • cgDataSource • cgDataSource-nonXA 	✓	✓	✓	✓
	portalFrameworkPool		✓		✓

Deployment Targeting Reference

Deployment Checklists

The following checklists are provided for each step in the deployment:

- [Checklist for Planning the Promotion](#)
- [Checklist for Installation and Network Configuration Requirements](#)
- [Checklist for Creating and Configuring a WebLogic Domain](#)
- [Checklist for Configuring a Production Database](#)
- [Checklist for Configuring Security](#)
- [Checklist for Using Load Balancers and Web Proxy Servers](#)
- [Checklist for Preparing Application Files](#)
- [Checklist for Deploying the Application](#)

Checklist for Planning the Promotion

The following checklist includes the items that you need to consider when planning the promotion of your application to different target environments. A plan should be prepared for each promotion stage; for example, the development, system integration, test/QA, and production environments.

Table B-1 Checklist—Planning the Promotion

Planning Item	Notes
<input type="checkbox"/> Identify machines and their hardware required to host WebLogic Server instances, the database, load balancers, firewalls, and so on.	<ul style="list-style-type: none"> Identify operating system and hardware requirements. Identify any OS patches required.
<input type="checkbox"/> Identify networking hardware and software requirements	Storage Area Network (SAN) or a multi-ported disk system is required for high availability in a cluster
<input type="checkbox"/> Define names, addresses, and port numbers of each machine	See “Guidelines for Specifying the Names and Addresses of the Members in a Cluster” on page 3-12.
<input type="checkbox"/> Define the multicast addresses for the cluster(s)	Ensure that the multicast address and port do not conflict with the multicast address and port of any other clusters on the network. This multicast address and port is used by cluster members to communicate within a cluster.
<input type="checkbox"/> Identify changes to the application source and configuration	<p>For example:</p> <ul style="list-style-type: none"> Source code changes required for the target environment, such as for enabling high availability or specific security requirements Deployment descriptor changes Configuration changes required for WebLogic Integration application views <p>Also identify application version control requirements; for example, identifying files to maintain in a version control tool such as Perforce or CVS.</p>

Table B-1 Checklist—Planning the Promotion (Continued)

Planning Item	Notes
<input type="checkbox"/> Security requirements	For example: <ul style="list-style-type: none"> • Required WebLogic security providers • LDAP server requirements • List of resources that need to be secured, including security policies required to protect them • SSL requirements, including certificates and how they are stored, identity of Certificate Authority, etc. • Action on Security advisories that are required
<input type="checkbox"/> Third-party components	For example, monitoring tools or LDAP
<input type="checkbox"/> Enterprise-level database software	Identify database access requirements with your database administrator
<input type="checkbox"/> JDBC database drivers	Determine whether a packaged JDBC database driver is to be used, or whether a third party driver is required; identify XA and non-XA requirements
<input type="checkbox"/> Load balancer or Web proxy server	Determine whether a hardware load balancer is to be used, or whether a third-party Web server or WebLogic Server is to be used as a proxy server
<input type="checkbox"/> Node Manager requirements	Determine whether Node Manager is to be used to administer Managed Servers, and whether it is to be started as a Web service or via a script
<input type="checkbox"/> SDK requirements	Determine whether the SDK is packaged with the WebLogic Platform installation or must be acquired separately. See <i>WebLogic Platform 8.1 Supported Configurations</i> at the following URL: http://e-docs.bea.com/platform/suppconfigs/configs81/81_over/overview.html
<input type="checkbox"/> Tools for configuration, application build, and deployment	For example, Configuration Wizard, WLST Offline, <code>wlwBuild</code> , <code>weblogic.deploy</code> , source control management system

Table B-1 Checklist—Planning the Promotion (Continued)

Planning Item	Notes
<input type="checkbox"/> Test plan, which specifies set of unit, stress, or other QA test requirements for each promotion stage	
<input type="checkbox"/> Script automation	Identify scripts to create and tools to use for promoting applications from development to production in an automated manner. See “Automating the Promotion Process” on page 1-8.

Checklist for Installation and Network Configuration Requirements

The following checklist includes the requirements for installing the software and configuring the network in a secure manner.

Table B-2 Checklist—Installation and Network Configuration Requirements

Requirements	Notes
<input type="checkbox"/> Procure the hardware and software, as defined in the promotion plan.	Operating system patches, service packs
<input type="checkbox"/> Install WebLogic Platform in a secure manner on all machines targeted for use in the domain.	<p>For details about installing WebLogic Platform, see <i>Installing WebLogic Platform</i> at the following URL: http://e-docs.bea.com/platform/docs81/install/index.html</p> <p>Follow the guidance described in “Install WebLogic Server in a Secure Manner” in “Determining Your Security Needs” in <i>Securing a Production Environment</i> at the following URL: http://e-docs.bea.com/wls/docs81/lockdown/secure.html</p> <p>Ensure the following:</p> <ul style="list-style-type: none"> • The software version and service pack are the same across all machines in the domain. • No machine has a dynamically assigned IP address. • All machines are accessible from the load balancer/proxy, and the load balancer/proxy is accessible by the clients. If machines are located behind a firewall, a public address must be available. • Each Managed Server is able to connect to the Administration Server in its domain. • A shared file system is not used to support a single installation running multiple WebLogic Server instances on separate machines. Doing so introduces a single point of contention for the cluster, as all server instances must compete to access the file system (and possibly to write individual log files). In addition, if the shared system fails, you might not be able to start server instances in the cluster.

Table B-2 Checklist—Installation and Network Configuration Requirements (Continued)

Requirements	Notes
<input type="checkbox"/> Procure a cluster production license for each product installation.	For licensing information, see <i>Licensing</i> at the following URL: http://e-docs.bea.com/platform/docs81/interm/license.html
<input type="checkbox"/> Set up the network.	Ensure that multicast traffic is enabled to support clustering and that IP sockets are configured correctly. Note that some network topologies can interfere with multicast communication. For more information, see “WebLogic Server Communication in a Cluster” in “Communications in a Cluster” in <i>Using WebLogic Server Clusters</i> at the following URL: http://e-docs.bea.com/wls/docs81/cluster/features.html
<input type="checkbox"/> Install the Web proxy plug-ins to support load balancing (Optional)	Obtain the latest WebLogic proxy plug-ins. This step is not required if you are using WebLogic Server as a proxy or a hardware load balancer.

Checklist for Creating and Configuring a WebLogic Domain

The following checklist includes the requirements for creating and configuring a WebLogic domain.

Table B-3 Checklist—Creating and Configuring a WebLogic Domain

Requirements	Notes
<input type="checkbox"/> Configure and target resources, including servers, clusters, machines, and JMS and JDBC resources.	<ul style="list-style-type: none"> • “Considerations for Configuring and Targeting Resources” on page 3-3 • “Guidelines for Specifying the Names and Addresses of the Members in a Cluster” on page 3-12 • “Autoconfiguration Using the Configuration Wizard and WLST Offline” on page 3-12
<input type="checkbox"/> Add asynchronous dispatcher queues required by the application, as defined in the <code>wlw-manifest.xml</code> file.	“Adding Application Resources Required by the WebLogic Workshop Runtime” on page 3-13
<input type="checkbox"/> Configure servers to start in production mode.	“Configuring Servers to Start in Production Mode” on page 3-14
<input type="checkbox"/> Set the SDK.	Set the SDK, as described in “Setting the SDK” on page 3-14
<input type="checkbox"/> Set up the Managed Servers on the remote machines and configure the start attributes.	<ul style="list-style-type: none"> • “Setting Up the Managed Server Directories” on page 3-15 • “Configuring the Managed Server Start Attributes” on page 3-16
<input type="checkbox"/> Configure Node Managers (Optional).	If Node Manager is to be used, see “Configuring Node Manager” on page 3-21

Checklist for Configuring a Production Database

The following checklist includes the requirements for configuring a production database.

Table B-4 Checklist—Configuring a Production Database

Requirements	Notes
<input type="checkbox"/> Create and prepare a production database.	<ul style="list-style-type: none"> • “Creating and Preparing a Production Database” on page 4-1 • “Example: How to Load the Domain Database Using WLST Offline” on page 4-6 • “Example: How to Load the Application Database Using Ant” on page 4-7
<input type="checkbox"/> Create conversational state tables, as defined in the <code>wlw-manifest.xml</code> file.	<ul style="list-style-type: none"> • “Creating Conversational State Database Tables” on page 4-2 • “Example: How to Create the Conversational State Database Tables Using Ant” on page 4-9
<input type="checkbox"/> Promote WebLogic Portal datasync information.	“Promoting WebLogic Portal Datasync Information” on page 4-3
<input type="checkbox"/> Promote LDAP and Portal database data.	“Promoting LDAP and Portal Database Data” on page 4-5
<input type="checkbox"/> Promote WebLogic Integration application database information.	“Promoting WebLogic Integration Application Database Information” on page 4-5

Checklist for Configuring Security

The following checklist includes the considerations for securing the production environment.

Table B-5 Checklist—Configuring Security

Considerations	Notes
<input type="checkbox"/> Configure a firewall.	<p>“Firewall Considerations” in “Avoiding Problems” in “Clustering Best Practices” in <i>Using WebLogic Server Clusters</i> at the following URL:</p> <p>http://e-docs.bea.com/wls/docs81/cluster/best.html</p>
<input type="checkbox"/> Use a load balancer or Web proxy server.	<p>A load balancer or Web proxy server distributes client connection requests, provides load balancing and failover across a WebLogic cluster, and provides security by concealing the local area network addresses from external users.</p> <p>For more information, see “Using Load Balancers and Web Proxy Servers” on page 6-1.</p>
<input type="checkbox"/> Secure the network connections.	<p>“Securing Network Connections” in “Ensuring the Security of Your Production Environment” in <i>Securing a Production Environment</i> at the following URL:</p> <p>http://e-docs.bea.com/wls/docs81/lockdown/practices.html</p>
<input type="checkbox"/> Secure the WebLogic Server hosts.	<p>“Securing the WebLogic Server Hosts” in <i>Securing a Production Environment</i> at the following URL:</p> <p>http://e-docs.bea.com/wls/docs81/lockdown/practices.html#SecuringWLSHost</p>
<input type="checkbox"/> Secure the WebLogic domain.	

Table B-5 Checklist—Configuring Security (Continued)

Considerations	Notes
<input type="checkbox"/> Secure WebLogic resources.	<p>For information about:</p> <ul style="list-style-type: none"> • The WebLogic Portal authentication framework, see “Securing Portal Applications” in <i>WebLogic Workshop Online Help</i> at the following URL: http://e-docs.bea.com/workshop/docs81/doc/en/portal/security/securityIntro.html • “Using Multiple Authentication Providers with WebLogic Portal” in <i>WebLogic Administration Portal Online Help</i> at the following URL: http://e-docs.bea.com/wlp/docs81/adminportal/help/SA_UsingMap.html • Configuring and promoting WebLogic Portal database tables, see “Creating and Preparing a Production Database” on page 4-1. • Using an external database store, see “Using an External Store for User Information” in <i>Security in WebLogic Platform 8.1</i> at the following URL: http://e-docs.bea.com/platform/docs81/secintro/user.html
<input type="checkbox"/> Secure WebLogic Portal resources.	<ul style="list-style-type: none"> • “Securing Portal Applications” in <i>WebLogic Workshop Online Help</i> at the following URL: http://e-docs.bea.com/workshop/docs81/doc/en/portal/security/securityIntro.html • “Using Multiple Authentication Providers with WebLogic Portal” in <i>WebLogic Administration Portal Online Help</i> at the following URL: http://e-docs.bea.com/wlp/docs81/adminportal/help/SA_UsingMap.html • “Creating and Preparing a Production Database” on page 4-1 • “Using an External Store for User Information” in <i>Security in WebLogic Platform 8.1</i> at the following URL: http://e-docs.bea.com/platform/docs81/secintro/user.html

Table B-5 Checklist—Configuring Security (Continued)

Considerations	Notes
<input type="checkbox"/> Secure WebLogic Integration resources.	<ul style="list-style-type: none"> • “Using WebLogic Integration Security” in <i>Deploying WebLogic Integration Solutions</i> at the following URL: http://e-docs.bea.com/wli/docs81/deploy/secure.html • “Promoting WebLogic Integration Application Database Information” on page 4-5
<input type="checkbox"/> Secure the database.	<ul style="list-style-type: none"> • “Ensuring the Security of Your Production Environment” in <i>Securing a Production Environment</i> at the following URL: http://e-docs.bea.com/wls/docs81/lockdown/practices.html • “Managing the Embedded LDAP Server” in <i>Managing WebLogic Security</i> at the following URL: http://e-docs.bea.com/wls/docs81/secmanage/ldap.html • “Using an External Store for User Information” in <i>Security in WebLogic Platform 8.1</i> at the following URL: http://e-docs.bea.com/platform/docs81/secintro/user.html
<input type="checkbox"/> Secure the application.	<ul style="list-style-type: none"> • “Preparing Application Security” on page 7-16 • “Securing Applications” in “Ensuring the Security of Your Production Environment” in <i>Securing a Production Environment</i> at the following URL: http://e-docs.bea.com/wls/docs81/lockdown/practices.html • <i>Programming WebLogic Security</i> at the following URL: http://e-docs.bea.com/wls/docs81/security/index.html
<input type="checkbox"/> Enable auditing.	<p>BEA recommends that you enable auditing in a production environment. An auditing provider stores operating requests and the results of those requests are collected, stored, and distributed for the purposes of non-repudiation. For more information about configuring auditing, see “Configuring Security Providers” in <i>Managing WebLogic Security</i>.</p>
<input type="checkbox"/> Review considerations for configuring security.	<p>“Considerations for Configuring Security” on page 5-6</p>

Table B-5 Checklist—Configuring Security (Continued)

Considerations	Notes
<input type="checkbox"/> Promote embedded LDAP security data to the target database.	“Promoting Embedded LDAP Security Data to the Target Database” on page 5-7
<input type="checkbox"/> Maintain the security policy files under version control.	“Maintaining Security Policy Files Under Version Control” on page 5-8

Checklist for Using Load Balancers and Web Proxy Servers

The following checklist includes requirements for using load balancers and Web proxy servers.

Table B-6 Checklist—Using Load Balancers and Web Proxy Servers

Requirements	Notes
<input type="checkbox"/> Configure the Web proxy server or external hardware load balancer.	“Load Balancing with a Web Proxy Server” on page 6-2
<input type="checkbox"/> Configure the external hardware load balancer.	“Load Balancing with an External Hardware Load Balancer” on page 6-1
<input type="checkbox"/> Review considerations when configuring load balancers and Web proxy servers, such as setting up replication groups.	“Considerations When Configuring Load Balancers and Web Proxy Servers” on page 6-3
<input type="checkbox"/> Configure load balancing and failover for servlets and JSPs, EJBs and RMIs, JMS destinations, and JDBC connections.	For more information see “Load Balancing in a Cluster” and “Failover and Replication in a Cluster” in <i>Using WebLogic Server Clusters</i> at the following URLs, respectively: <ul style="list-style-type: none"> • http://e-docs.bea.com/wls/docs81/cluster/load_balancing.html • http://e-docs.bea.com/wls/docs81/cluster/failover.html

Checklist for Preparing Application Files

The following checklist includes the requirements for preparing the application files.

Table B-7 Checklist—Preparing Application Files

Requirements	Notes
<input type="checkbox"/> Obtain application files.	“Access to Application Files” on page 7-7
<input type="checkbox"/> Update application project files to reference the domain in the target environment on which the application runs.	“Updating the Server Path Attribute” on page 7-18
<input type="checkbox"/> Update WebLogic Portal cache configuration, behavior tracking, campaign, and commerce tax settings.	See instructions about updating the <code>application-config.xml</code> file in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3
<input type="checkbox"/> Tune the application for a production environment.	See instructions for modifying the <code><jsp-descriptor></code> element of the <code>weblogic.xml</code> deployment descriptor in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3.
<input type="checkbox"/> Tune WebLogic Workshop dispatcher and container EJBs for a production environment.	See instructions for modifying these EJBs in “Summary of Changes to Deployment Descriptors and Configuration Files” on page 7-3.
<input type="checkbox"/> Enable session replication to maintain information about object state in the event of a server failure.	Configure the <code><session-param></code> descriptor element in the <code>weblogic.xml</code> deployment descriptor file, as described in “Enabling Session Replication” on page 7-14
<input type="checkbox"/> Update host, port, and optionally security protocol of target service specified in the Web service controls.	“Enabling Inter-Application Communication” on page 7-11
<input type="checkbox"/> Set the concurrency strategy for stateful business process entity beans.	Make sure that the <code><ejb-concurrency-strategy></code> element in the <code>wlw-config.xml</code> file is set to exclusive for stateful business process entity beans, as described in “Specifying a Concurrency Strategy for Stateful Business Process Entity Beans” on page 7-14

Table B-7 Checklist—Preparing Application Files (Continued)

Requirements	Notes
<input type="checkbox"/> Update the WebLogic Integration application views and adapter target event generators.	<ul style="list-style-type: none">• “Reconfiguring Application Views” on page 7-15• “Specifying Event Generator Targets” on page 7-16
<input type="checkbox"/> Set application security	<ul style="list-style-type: none">• “Limiting Access to Authenticated Users” on page 7-16• “Restricting Application Access to SSL Traffic Only” on page 7-17

Checklist for Deploying the Application

The following checklist includes requirements for deploying the application.

Table B-8 Checklist—Deploying the Application

Requirements	Notes
<input type="checkbox"/> Start the servers.	“Starting the Servers” on page 8-9
<input type="checkbox"/> Upload the deployment files to the Administration Server (Optional).	“Uploading Deployment Files to the Administration Server” in “Performing Common Deployment Tasks” in <i>Deploying WebLogic Server Applications</i> at the following URL: http://e-docs.bea.com/wls/docs81/deployment/scenarios.html
<input type="checkbox"/> Deploy the application.	<ul style="list-style-type: none"> • “Overview of the Deployment Tools” on page 8-2 • “Deployment Considerations” on page 8-3 • “Step 3: Deploy the Application” on page 8-8
<input type="checkbox"/> Deploy Event Generators (Optional).	“Creating and Deploying Event Generators” in “Event Generators” in <i>Managing WebLogic Integration Solutions</i> at the following URL: http://e-docs.bea.com/wli/docs81/manage/evntgen.html