BEA WebLogic Portal™

Deployment Guide

Version 4.4
Document Date: May 2002

Deployment Guide

Document Edition	Date	Software Version
4.4	April 2002	WebLogic Portal 4.0, Service Pack 2
4.3	February 2002	WebLogic Portal 4.0, Service Pack 1
4.1	November 2001	WebLogic Portal 4.0 with support for Microsoft SQL Server
4.0	October 2001	WebLogic Portal 4.0

Contents

About This Document

This document guides you through assembling your organization's BEA WebLogic Portal^{TM} Web application and enterprise application, deploying the applications, and synchronizing data between the E-Business Control Center and the applications.

In addition, this document describes setting up databases for the WebLogic Portal data repository. It includes the following topics:

- Chapter 1, "Deployment Overview," which provides an overview of the process of deploying application servers and applications.
- Chapter 2, "Files for the Application Server," which describes reviewing default settings used to deploy WebLogic Portal.
- Chapter 3, "Deploying Domains and Server Configurations," which describes reviewing default settings for the WebLogic Portal domain and enterprise application.
- Chapter 4, "Assembling Your Web Application," which describes requirements for assembling your e-business Web application.
- Chapter 5, "Assembling and Deploying Enterprise Applications," which describes packaging your Web application, EJBs, and other supporting Web applications as an enterprise application.
- Chapter 6, "Deploying Clusters," which guides you through creating and deploying a cluster. A WebLogic Server *cluster* is a group of WebLogic Server instances that work together to provide a powerful and reliable Web application platform. A cluster appears to its clients as a single server but it is, in fact, a group of servers acting as one. It provides two key benefits that are not provided by a single server: scalability and availability.
- Chapter 7, "Synchronizing Application Data," which describes how to deploy application data, such as definitions for campaigns and customer segments, to your enterprise application.

- Chapter 8, “Starting and Shutting Down a Server,” which describes configuration and startup files for WebLogic Portal.
- Chapter 9, “Configuring WebLogic Portal for Oracle Databases,” which describes creating the WebLogic Portal database schemas for Oracle databases.
- Chapter 10, “Configuring WebLogic Portal for Microsoft SQL Server Databases,” which describes creating the WebLogic Portal database schemas for Microsoft SQL Server databases.

What You Need to Know

This document is intended mainly for Application Assemblers/Deployers, System Administrators, and Database Administrators. It assumes a familiarity with WebLogic Portal, the WebLogic Server platform, J2EE specifications, as well as the database management system that your organization uses.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at http://e-docs.bea.com.

How to Print the Document

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

A PDF version of this document is available on the WebLogic Portal documentation Home page on the e-docs Web site. A PDF version of this document is also available in the documentation kit on the product CD. Or you can download the documentation kit from the WebLogic Portal portion of the BEA Download site. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Portal documentation Home page, click the PDF files button and select the document you want to print.

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

Related Information

The following documents provide background and additional information that you may need to deploy WebLogic Server and WebLogic Portal:

- *Java™ 2 Platform Enterprise Edition Specification, v1.3*
- *BEA WebLogic Server Administration Guide*
- *Developing WebLogic Server Applications*
- *Performance Tuning Guide*

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Portal **Product Version: 4.0** release.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support through BEA WebSUPPORT at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.

Convention	Item
italics	Indicates emphasis or book titles.

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

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

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

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

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

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

Part I Deploying a Server and Enterprise Application

CHAPTER

1 Deployment Overview

You can deploy your e-business Web site into a variety of environments with unique network topographies. For example, you can combine WebLogic Portal with proxy servers, load balancers and firewalls to meet your security and performance requirements.

This document describes how to deploy WebLogic Portal servers, enterprise applications, and application data. Table 1-1 describes the components of a WebLogic Portal Web site and indicates the data structures in which the components are deployed.

Table 1-1 Deployment Components

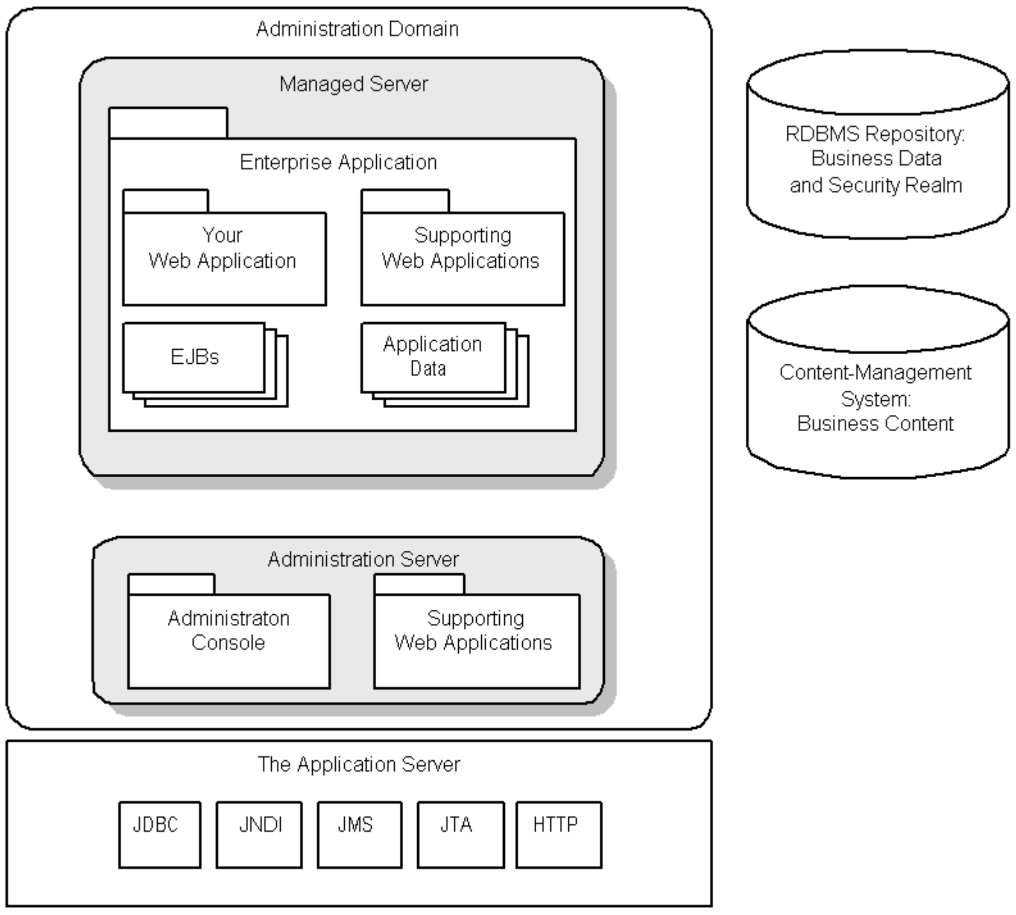
This Component	Provides	Is Deployed As
Application Server	JDBC, JNDI, JMS, JTA, security, HTTP, and other essential services. It is the layer between the Java Virtual Machine platform and the enterprise application.	A set of Java classes, license files, and other files under `bea/wlserver6.1` and `bea/wlportal4.0`. For more information, refer to Chapter 2, “Files for the Application Server.”
Domain and Servers	Configurations for the application server. You can create multiple configurations and use each one to start a separate server instance. A domain also defines clusters, which group servers for the purposes of load balancing and failover. Usually, you configure your domain to contain an Administration Server and one or more Managed Servers.	A directory that contains a `config.xml` file, optional security certificates, and one or more applications. For more information, refer to Chapter 3, “Deploying Domains and Server Configurations.”

Table 1-1 Deployment Components (Continued)

This Component	Provides	Is Deployed As
Enterprise Application and Web Application	Management of your customer's experience on your Web site. For example, applications provide portals and portlets, present content and product information, engage customers in promotional campaigns, and process purchases.	A directory tree, either expanded or archived into a single file. For more information, refer to Chapter 4, "Assembling Your Web Application," and Chapter 5, "Assembling and Deploying Enterprise Applications."
Application Data	Expresses your business model. It specifies how an application behaves to support your business goals. For example, portlet properties, entitlement segments, rules, Webflows, Pipelines, and campaigns are all expressed as application data.	XML files within a domain. You use the E-Business Control Center to make this data available to an application. For more information, refer to Chapter 7, "Synchronizing Application Data."
Business Data	Representations of customers, products, and orders. The Customer Registration, Payment, and other services generate this data. For example, User profiles, product catalog, orders and order histories are business data.	Data in the RDBMS repository. You use the WebLogic Portal Administration Tools, database-vendor tools, third-party management tools, and other scripts that the server supplies to manage the data in the RDBMS repository. For more information, refer to Part II "Deploying Your Business Data."
Content	The information that you present to your customers. You can add content, such as HTML fragments, directly to your JSPs or use the JSPs to retrieve documents from a content-management system.	Files or data stored in a content management system. WebLogic Portal includes a simple content management system along with integrations with third-party systems. The default content management system stores data in the RDBMS repository. For information about deploying third-party content management systems, refer to the documentation from the third-party vendor.
Security Realm	Authentication of your developers and customers.	Data in the RDBMS repository, if you use the default wlcsRealm or some other RDBMS Realm. You can use other realms, such as LDAP realm, which have other deployment requirements.

Figure 1-1 illustrates a fully deployed Web site.

Figure 1-1 Overview of a Deployed Site



2 Files for the Application Server

When you install WebLogic Server and WebLogic Portal, the installation programs set up a fully configured application server, sample applications and supporting EJBs, and other files that you use as a reference point for developing your enterprise application.

These reference resources are appropriate for installing in a development environment. In a production environment, if you do not want to use all of the files as they are organized by the installation programs, refer to the sections of this topic:

- Required Files and Directories
- Recommended Files and Directories
- Support for Developing Applications

Required Files and Directories

You must make the files and directories listed in Listing 2-1 available to any environment in which you want to run a WebLogic Portal application server.

Table 2-1 Required Files and Directories

Required Component	Description
The `BEA_HOME` directory	Named `bea` by default. For information about the files in the `BEA_HOME` directory, refer to "Preparing to Install WebLogic Server" in the *WebLogic Server Installation* guide. You must specify the name and location of `BEA_HOME` when you start the application server. For more information, refer to Chapter 8, "Starting and Shutting Down a Server."
The JDK	By default, the WebLogic Server installer includes the required JDK version. For information on the required version, refer to "Supported Platforms" in the *Installation Guide*.
The `BEA_HOME/wlserver6.1` directory	Contains files and directories for running the application server. The WebLogic Server installer includes an option for installing only required files.
`PORTAL_HOME/lib`	Provides JARs that extend WebLogic Server with WebLogic Portal features. Some of the JARs in this directory must be named in the `CLASSPATH` environment variable. (For more information, refer to "Add Directories to CLASSPATH" on page 8-5.) Do not modify any of these files and directories.
`PORTAL_HOME/weblogiccommerce.properties`	A file that configures domains and servers (in addition to standard J2EE deployment descriptors). It is deprecated along with the sections of Java code that it configures.

Recommended Files and Directories

Listing 2-2 lists files and directories that are not required. However we recommend that you include them in an environment to facilitate installing service packs, upgrading, and starting and stopping a server instance.

Table 2-2 Recommended Files and Directories

Recommended Component	Description
`BEA_HOME/registry.xml`	Contains a persistent record of all BEA products installed on the target system. This registry contains product-related information, such as version level, Service Pack level, and installation directory. It facilitates the installation of Service Packs and upgrades.
`BEA_HOME/utils/*`	Contains utilities that are used to support the installation of all BEA WebLogic Server products. The `utils.jar` file contains code that supports the `UpdateLicense` utility. It facilitates the installation of Service Packs and upgrades.
`PORTAL_HOME/` `StartPortal` and `StopPortal`, or their equivalents	Scripts that start and stop WebLogic Portal domains. For more information, refer to Chapter 8, "Starting and Shutting Down a Server."
`PORTAL_HOME/bin/` `set-environment.bat` (or `.sh`)	A script that sets all environment variables required for the run-time environment. For more information, see "Setting Environment Variables" on page 8-4.
`PORTAL_HOME/` `uninstaller`	A directory that contains files for uninstalling WebLogic Portal.

Support for Developing Applications

WebLogic Portal installs the files and directories in Listing 2-3 to support your development efforts. You do not need to deploy them in a production environment.

Table 2-3 Files and Directories that Support Development

Development Component	Description
`PORTAL_HOME/config`	The domain repository (parent directory) for the sample WebLogic Portal domains. The config directory that the installer creates is for development purposes. Your production environment must include a config directory, but it does not need to contain the sample domains, and it does not need to be located below `PORTAL_HOME`. For more information, see Chapter 3, “Deploying Domains and Server Configurations.”
`PORTAL_HOME/applications`	A directory that contains sample enterprise applications. You can deploy your enterprise applications in this or other directories. For more information, refer to Chapter 5, “Assembling and Deploying Enterprise Applications.”
`PORTAL_HOME/bin`	A directory that contains platform-specific scripts for setting up your environment.
`PORTAL_HOME/db`	A directory that contains the Cloudscape database.Use this database for demonstration purposes only. To see a list of databases that we support for development and production, refer to Supported Platforms in the *Installation Guide*. This directory also contains scripts for creating and migrating schemas for development and production databases, and for loading those databases with sample data. You might need to modify some of these scripts, depending on your environment. We recommend that you make a backup copy of the entire directory tree or place it under source control before you modify it.
`PORTAL_HOME/dmsBase`	Directories that contain content to support the sample enterprise applications.

CHAPTER

3 Deploying Domains and Server Configurations

A WebLogic Server *domain* is the administrative unit that you use to manage servers and enterprise applications. A *server configuration* is a set of properties that determines operating parameters for an active instance of an application server.

Each domain contains one or more active servers. For any given development project, you might create several different domain and server combinations, each configured to support the following types of environments:

- Development environments. You might create multiple development domains for supporting different aspects of your site's development. For example, most developers would work in a single-server environment, but your System Administrator might create a separate domain for developing and configuring a cluster.
- Testing environments. You might configure several testing domains to test different load balancing scenarios.
- Production environments.

WebLogic Portal includes sample (reference) domains, which are configured to support a single-server development environment.

The following sections in this topic describe creating and deploying domains:

- How WebLogic Portal Domains Are Organized
- Creating Domains and Configuring Servers
- Place the New Directory Tree Under Source Control

How WebLogic Portal Domains Are Organized

WebLogic Portal domains conform to all requirements that WebLogic Server imposes for administrative domains. While they do not extend domain functionality or introduce additional requirements, they do represent one of many possible implementations. The files and directories in Table 3-1 are required for all WebLogic Portal domains.

Note: WebLogic Portal installs its reference domains in `PORTAL_HOME/config`.

Table 3-1 Files and Directories in a WebLogic Portal Domain

Component	Description
`config/`*domain*`/config.xml`	The persistent storage for the domain's configuration information. Among other things, `config.xml` stores the list of EJBs, web applications, and enterprise applications that are in the domain. The WebLogic Server Administration Console creates and maintains this file. Do not use any other tool to modify the `config.xml` file for an active domain.
`config/`*domain*`/fileRealm.properties`	Describes ACLs and an encrypted version of the password needed to start the Administration Server for the domain. Even if you use some other security realm to authenticate users, WebLogic Portal uses `fileRealm.properties` to describe the administrator ACL and password. The `SerializedSystemIni.dat` file must accompany `fileRealm.properties` in any domain.

Table 3-1 Files and Directories in a WebLogic Portal Domain (Continued)

Component	Description
`config/`*`domain`*`/logs`	A directory that contains WebLogic Server log files. For more information, refer to "Using Log Messages to Manage WebLogic Servers" in the *BEA WebLogic Server Administration Guide*.
`config/`*`domain`*`/*.pem`	Security certificates for mutual authentication. If your enterprise application uses mutual authentication, each domain in which you deploy the application must contain its own certificates.

For information on how WebLogic Server organizes domains and servers in general, refer to "Overview of WebLogic Server Management" in the *WebLogic Server Administration Guide*. For additional background information, refer to the "Application Assembly and Deployment" chapter of the Java™ 2 Platform Enterprise Edition Specification, v1.3.

Creating Domains and Configuring Servers

We recommend that you use one of the reference domains and servers as a starting point for your development.

This section contains the following subsections:

- Review the Reference Domains
- Create Your Domains
- Configure Servers
- Add Extensions to the WebLogic Server Administration Console
- Make weblogiccommerce.properties Available to the Domain
- Specify a Default Web Application
- Relocate the Domain (Optional)

Review the Reference Domains

Review the following domains that WebLogic Portal provides and choose the one that most closely resembles your business needs:

- petflowDomain
- p13nDomain
- wlcsDomain
- portalDomain

petflowDomain

petflowDomain contains the petflowApp application, which features a demonstration of the Webflow service. To activate the domain and server, do the following:

1. Change to the `PORTAL_HOME/config/petflowDomain` directory.
2. Run `startPetflow.bat` (`startPetflow.sh` on UNIX).

Then, to view petflowApp, in a Web browser enter the following URL:
`http://localhost:7501/petflow`

p13nDomain

p13nDomain contains the p13nApp application, which demonstrates personalization services. To activate the domain and server, do the following:

1. Change to the `PORTAL_HOME/config/p13nDomain` directory.
2. Run `startP13N.bat` (`startP13N.sh` on UNIX).

Then, to view p13nApp from the p13nDomain, in a Web browser enter the following URL:
`http://localhost:7501/`

Note that this URL only specifies the server name and port number because the p13n Web application within p13nApp has been configured as the default Web application of the p13nDomain.

Any domain can specify a default Web application, which responds to HTTP requests that cannot be resolved to another deployed Web application. For information on how to specify a default Web application, refer to “Configuring WebLogic Server Web Components” in the *WebLogic Server Administration Guide*.

wlcsDomain

wlcsDomain contains both the p13nApp and wlcsApp applications. The p13nApp is the same application that you can access from p13nDomain. WebLogic Portal includes it in wlcsDomain to demonstrate that you can deploy multiple applications in a domain, and as a convenience to you: if you know that you want to use personalization and commerce services on your site, you can start wlcsDomain to access both p13nApp and wlcsApp.

wlcsApp features a demonstration of personalization, commerce, and campaign services.

To activate the domain and server, do the following:

1. Change to the `PORTAL_HOME/config/wlcsDomain` directory.
2. Run `startWLCS.bat` (`startWLCS.sh` on UNIX).

To view wlcsApp, in a Web browser enter the following URL:
`http://localhost:7501/wlcs`

Then, to view p13nApp, in a Web browser enter the following URL:
`http://localhost:7501/p13n`

Note that the p13n Web application is not the default Web application for wlcsDomain and therefore the URL must specify the context root of the p13n Web application. To view the default application for wlcsDomain, enter the following URL:
`http://localhost:7501/`

portalDomain

portalDomain includes p13nApp, wlcsApp, and portal. The portal application demonstrates the portal service.

Because portalDomain includes three applications, it provides the best demonstration of the scope of WebLogic Portal features. This document assumes that in your development environment, you use portalDomain as your reference, though you can use any domain that best reflects your business needs.

Each application that you add to a domain increases the amount of time to start the server.

To activate the portalDomain and server, do any of the following:

- From `PORTAL_HOME/config/portalDomain`, run `startPortal.bat` (`startPortal.sh` on UNIX).
- From `PORTAL_HOME`, run `StartPortal.bat` (`StartPortal.sh` on UNIX).

  `StartPortal.bat` calls `config/startPortal.bat`. We provide this script as a convenience to you.
- On Windows, click Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Portal 4.0 → Start BEA WebLogic Portal

Then, to view p13nApp, in a Web browser enter the following URL:
`http://localhost:7501/p13n`

To view wlcsApp, in a Web browser enter the following URL:
`http://localhost:7501/wlcs`

To view the portal application, in a Web browser enter the following URL:
`http://localhost:7501/stockportal`

Create Your Domains

To create a domain, do the following:

1. Install WebLogic Portal on a computer. Be sure to install the sample enterprise applications (which is the default installation configuration).
2. Use the `StartPortal.bat` script (`StartPortal.sh` on UNIX) to activate the portalDomain reference domain. On Windows, you can also activate this domain from the Start menu. For more information, refer to Chapter 8, “Starting and Shutting Down a Server.”

3. Access the WebLogic Server Administration Console for portalDomain by doing one of the following:

   - From the computer on which you installed and started the server, enter the following URL in a Web browser:

     `http://localhost:7501/console`

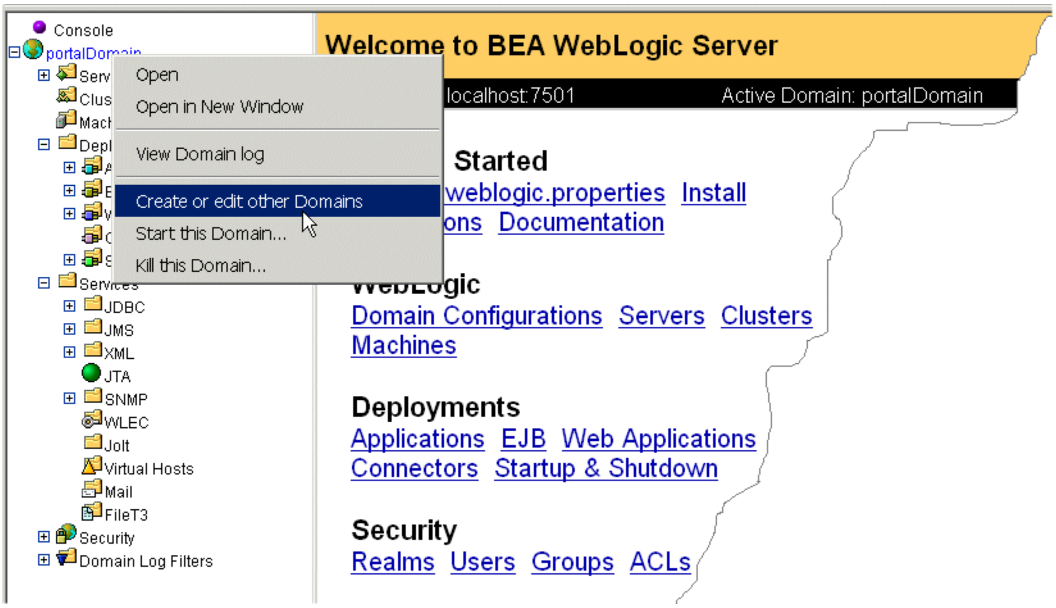
   - From a remote computer, enter the following URL in a Web browser:

     `http://`*`host-name or IP-address`*`:7501/console`

   Enter the system user ID and password as entered when you installed WebLogic Portal.

4. In the left pane of the WebLogic Server Administration Console, right click portalDomain. From the shortcut menu, click Create or Edit Other Domains. (See Figure 3-1.)

Figure 3-1 Create a Domain



5. In the All Domain Configurations window, in the Name box, enter a name for your domain. Use a name that indicates the purpose of the domain. For example, in an environment for developing applications for an online bank use `bank_development`. WebLogic Server creates a directory of the same name, so your domain name must satisfy naming requirements for the host file system.

6. Click Create.

   Under the `config` directory that contains the currently active domain (`portalDomain`), the WebLogic Server Administration Console does the following:

   a. Creates a directory for your domain. By default, the pathname for the directory is as follows:

      PORTAL_HOME/config/*name-of-your-domain*

      For example, if you named your domain `bank_development`, the WebLogic Server Administration Console creates the following directory:

      ```
      PORTAL_HOME/config/bank_development
      ```

   b. Under the new domain directory, the WebLogic Server Administration Console creates a `config.xml` file.

Configure Servers

This section guides you through setting up a server instance that supports WebLogic Portal enterprise applications:

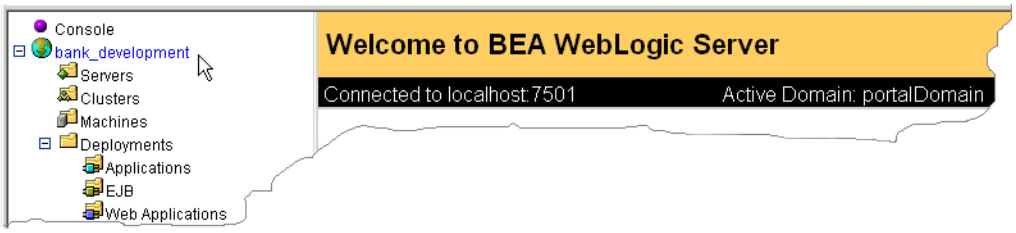
- Create a Server
- Configure Listen Ports
- Configure the Message Output
- Set Up a Security Realm
- Configure an XML Registry
- Set Up JDBC Connection Pools and Data Sources
- Viewing the JNDI Tree

Create a Server

To create a server instance, do the following:

1. In the WebLogic Server Administration Console, make sure you are editing your new domain by verifying that in the left pane, the top node of the tree states the name of your new domain. For example, in Figure 3-2, the console is editing the `bank_development` domain.

Figure 3-2 Editing the bank_development Domain



   If the top pane does not state the name of your new domain, do the following:

   a. In the left pane, right click the name of the current domain.

   b. From the short cut menu, click Create or Edit Other Domains. (See Figure 3-1.)

   c. On the All Domain Configurations page, click the name of your domain.

   Note that at this point, while you are editing your new domain, the `portalDomain` is still the **active** domain.

2. In the left pane, right click the Servers folder. From the shortcut menu, click Configure a new Server.

3. In the Create a New Server window, in the Name box, enter a name for the server. Use a name that indicates the purpose of the server instance. For example, for your development team, you can create a server named `bankServer`.

4. Click Create.

Configure Listen Ports

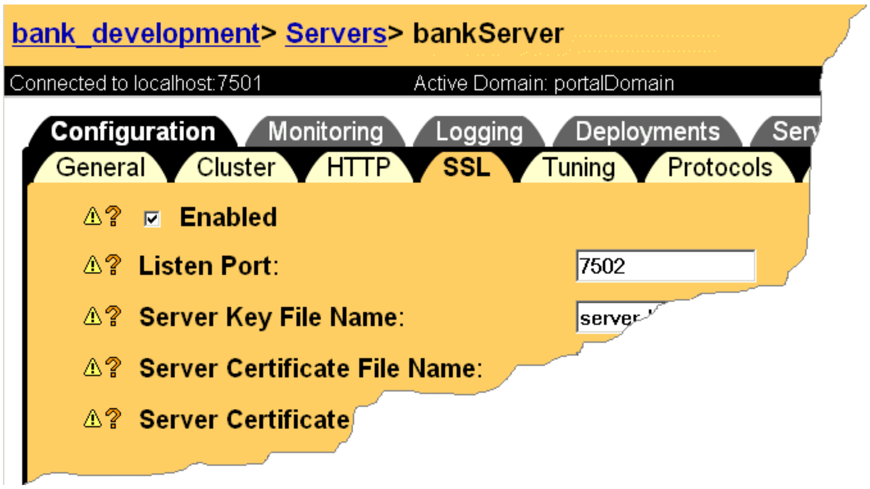
To change the default listen ports, do the following:

1. In the Create a New Server window, in the Listen Port box, change the value. Then click Apply. This is the listen port that you use to access the WebLogic Server Administration Console for your new domain. For example, if you change the port number to 7501, then you use the following URL to access the WebLogic Server Administration Console for your new domain:

   ```
   http://localhost:7501/console
   ```

2. Click the SSL tab.
3. Click the Enabled check box.
4. In the Listen Port box, change the value. (See Figure 3-3.)

Figure 3-3 The SSL Listen Port to Support Sample Applications


bank_development> Servers> bankServer
Connected to localhost:7501
Active Domain: portalDomain
Configuration
Monitoring
Logging
Deployments
Serv
General
Cluster
HTTP
SSL
Tuning
Protocols

☑ Enabled
Listen Port: 7502
Server Key File Name: server
Server Certificate File Name:
Server Certificate

5. On the SSL tab, enter the names of your security certificate files.

   You need a private key and digital certificate for each deployment of WebLogic Server that will use the SSL protocol. To acquire a digital certificate from a certificate authority, you must submit your request in a particular format called a Certificate Signature Request (CSR). For more information, refer to "Managing Security" in the *WebLogic Server Administration Guide*.

Each WebLogic Portal reference domain provides sample security files, which you can use for demonstration purposes in your domain. To use these files, copy all `*.pem` files from `PORTAL_HOME/config/portalDomain` to `PORTAL_HOME/config/`*`yourDomain`*. Then change the values on the SSL tab to specify the files that you copied.

6. Click Apply.

Note: You can create multiple servers in each domain. For example, you might create one server to support the enterprise applications and another server with different listen ports to support other applications.

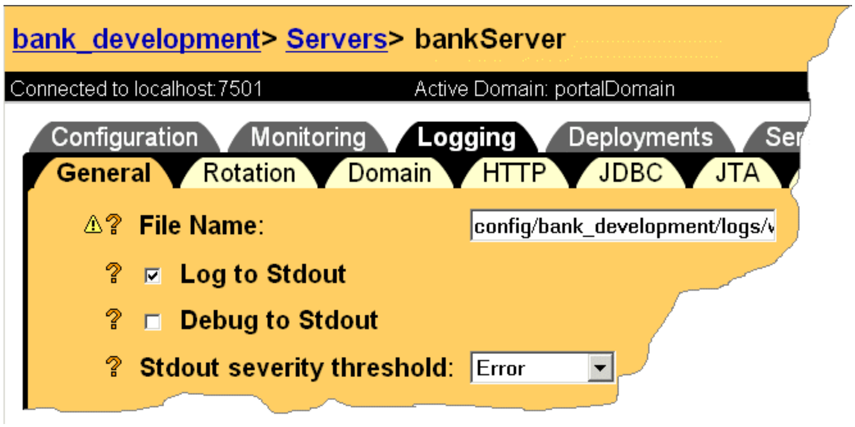
Configure the Message Output

You can use the WebLogic Server Administration Console to change the level of messages that the server generates and where the server locates log files.

To change the message output, do the following:

1. On the server properties page, click the Logging tab.
2. On the General subtab, select a value from the Stdout severity threshold list (see Figure 3-4). For more information about message and log options, refer to the WebLogic Server Administration Console Online Help.

Figure 3-4 Stdout Severity Threshold List



bank_development> Servers> bankServer

Connected to localhost:7501 Active Domain: portalDomain

File Name: config/bank_development/logs/v

☑ Log to Stdout

☐ Debug to Stdout

Stdout severity threshold: Error

3. To change the location in which the server locates its log files, modify the path and file name in the File Name box. Then click Apply.
4. To change the location in which the domain locates its log files, do the following:
    a. In the left pane of the WebLogic Server Administration Console, click your domain.
    b. In the right pane, click the Logging tab.
    c. In the File Name box, provide a new path and file name for the domain logs. Then click Apply.

Set Up a Security Realm

A security realm determines how a user is authenticated and retrieves access control lists for given names. The reference domain provides a custom security realm, `wlcsRealm`, that stores user IDs and passwords in the RDBMS repository. You can set up other types of security realms, as described in “Managing Security” in the *WebLogic Server Administration Guide*.

The reference domain also provides a caching realm, `wlcsCachingRealm`, which improves the performance of WebLogic Server by caching lookups, thereby reducing the number of calls into other security realms.

To set up `wlcsRealm` and `wlcsCachingRealm`, do the following:

1. In the left pane, expand the Security folder.
2. Right click the Realms folder, and click Configure a new RDBMSRealm.
3. On the Configure a new RDBMSRealm page, click the Configuration tab and enter the following values:

In this box...	**Enter this value...**
Name	`wlcsRealm`
Realm Class Name	`com.bea.p13n.security.realm.RDBMSRealm`

4. Click Create.

5. To use the sample Cloudscape database as the RDBMSRealm host, click the Database tab and enter the following values

In this box...	Enter this value...
Driver	`COM.cloudscape.core.JDBCDriver`
URL	`jdbc:cloudscape:demo;create=true;autocommit=false`
User Name	empty

   For information on using other database types as the RDBMSRealm host, refer to Part II, "Deploying Your Business Data."

6. Click Apply.
7. In the left pane, right click the Caching Realms folder and click Configure a new Caching Realm.
8. On the Configure a new Caching Realm page, click the Configuration tab and enter the following values:

In this box...	Enter this value...
Name	`wlcsCachingRealm`
Basic Realm	`wlcsRealm`
Case Sensitive Cache	selected

9. Click Create.

Start Your Server

The remaining configuration tasks require that you deploy the WebLogic Server Administration Console to your new domain and then start your server:

1. To stop the portalDomain, change to the `PORTAL_HOME` directory and run `stopPortal.bat` (`stopPortal.sh` on UNIX).

2. Copy the following files and directories from `PORTAL_HOME/config/portalDomain/` to `PORTAL_HOME/config/`*`myDomain/`*:
   - `fileRealm.properties`
   - `SerializedSystemIni.dat`
   - `democert.pem` and `demokey.pem`
3. To start your new domain and server, do the following:

   a. Copy `PORTAL_HOME/config/portalDomain/startPortal.bat` to `PORTAL_HOME/config/`*`myDomain`*`/start`*`MyDomain`*`.bat`.

   b. Open `start`*`MyDomain`*`.bat` in a text editor.

   c. Change the value for the `DOMAIN_NAME` to the name of your new domain. For example, `SET DOMAIN_NAME=bank_development`

   d. Change the value for the `SERVER_NAME` variable to the name of your new server. For example, `SET SERVER_NAME=bankServer`

   e. Run `start`*`MyDomain`*`.bat`.

   The shell in which you run the script displays WebLogic Server messages. The following is an example of the messages that indicate a successful startup:

```
<Oct 15, 2001 9:20:13 AM MDT> <Notice> <Management>
<Application Poller not started for production server.>

<Oct 15, 2001 9:21:49 AM MDT> <Notice> <WebLogicServer>
<ListenThread listening on port 7501>

<Oct 15, 2001 9:21:49 AM MDT> <Notice> <WebLogicServer>

<SSLListenThread listening on port 7502>

<Oct 15, 2001 9:21:51 AM MDT> <Notice> <WebLogicServer>
<Started WebLogic Admin Server "bankServer" for domain
"bank_development" running in Production Mode>
```

4. To access the WebLogic Server Administration Console for your new domain, do the following

   a. In the shell that is running the server that you started in step 3, note the port number in the following statement:
   ```
   <Oct 3, 2001 5:11:41 PM EDT> <Notice> <WebLogicServer>
   <ListenThread listening on port 7501>
   ```

   b. In a Web browser, enter the following URL:
   `http://localhost:`*`port-number`*`/console`

Configure an XML Registry

The XML registry configures XML parsers and XSLT transformers for your applications. For more information about the XML registry, refer to the "*WebLogic Server Programming WebLogic XML*" guide.

To configure an XML registry for your applications, do the following:

1. In WebLogic Server Administration Console, in the left pane, click Services.
2. Right click the XML folder. Then select Configure a New XML Registry.
3. On the Configure a new XML Registry page click the Configuration tab and enter the following values:

In this box...	**Enter this value...**
Name	`portal XML registry`
DocumentBuilderFactory	`weblogic.apache.xerces.jaxp.DocumentBuilderFactoryImpl`
SAXParserFactory	`weblogic.apache.xerces.jaxp.SAXParserFactoryImpl`
Transformer Factory	`weblogic.apache.xalan.processor.TransformerFactoryImpl`
When To Cache	cache-on-reference

4. Click Create.

5. On the portal XML registry page, click Configure XML Parser Select Registry Entries.

6. On the XML Parser Select Registry Entries page, click Configure a new XML Parser Select Registry Entry.

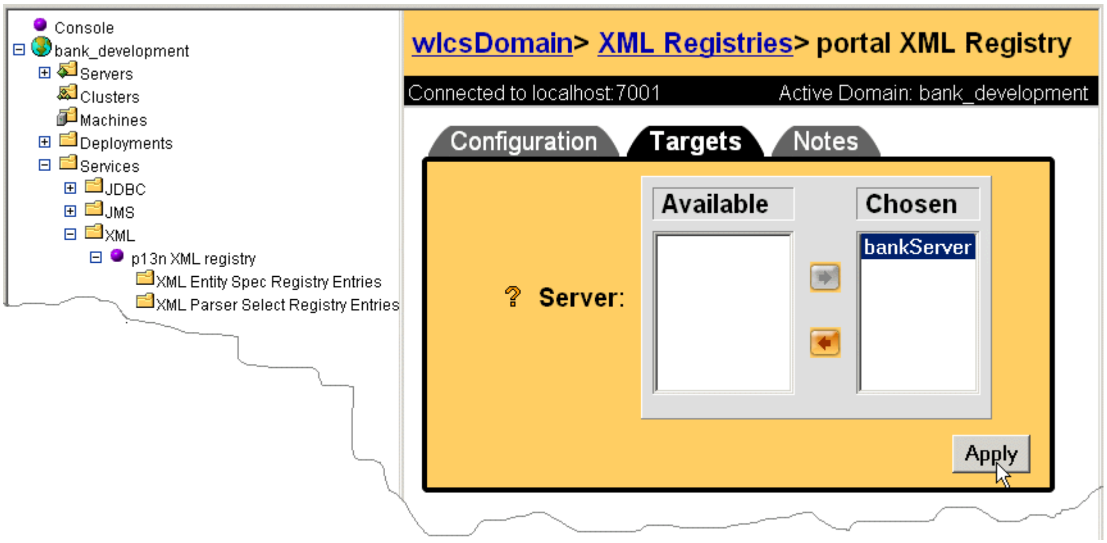
7. On the Create a new XMLParserSelectRegistryEntry page, in the SAXParser Factory box, enter the following value:
   `weblogic.apache.xerces.jaxp.SAXParserFactoryImpl`

8. Click Create.

9. To deploy the XML registry, in the left pane, click portal XML registry.

10. On the portal XML registry page, click the Targets tab. Then move your server from the Available to the Chosen list. (See Figure 3-5).

Figure 3-5 Deploy the XML Registry



Connected to localhost:7001 Active Domain: bank_development

Server: bankServer

Set Up JDBC Connection Pools and Data Sources

Connection pools provide ready-to-use pools of connections to your RDBMS. The application server creates the pools during server startup, thus eliminating the overhead of your enterprise application having to establish database connections for each transaction. For more information about connection pools, refer to "Overview of Connection Pools" in the *WebLogic Server Programming WebLogic JDBC* guide.

A DataSource is an interface between your enterprise application and a connection pool. For more information, refer to "Overview of DataSources" in the *WebLogic Server Programming WebLogic JDBC* guide.

This section contains the following subsections:

- Configure commercePool for Cloudscape Databases
- Tip: Cloning Resources
- Configure dataSyncPool for Cloudscape Databases
- Configure JDBC Data Sources

For information on setting up JDBC connection pools and data sources for your production RDBMS, refer to Part II, "Deploying Your Business Data."

Configure commercePool for Cloudscape Databases

Create a JDBC connection pool named `commercePool` for Cloudscape by doing the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Click the Connection Pools folder and click Configure a new JDBC Connection Pool. On the Configure a new JDBC Connection Pool page click the General tab. In the Name box, enter `commercePool`.

3. To configure commercePool, do the following:

   a. On the General tab, enter the following values:

In this box...	Enter this value...
URL	`jdbc:cloudscape:Commerce;create=true;upgrade=true`
Driver Classname	`COM.cloudscape.core.JDBCDriver`

Properties	`user=none`
	`password=none`
	`weblogic.t3.waitForConnection=true`
	`weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.waitSecondsForConnectionSecs=999999999999,verbose=false`
	`server=none`

   b. Click Create.

   c. Click the Connections tab and enter the following values:

In this box...	Enter this value...
Initial Capacity	`20`
Maximum Capacity	`20`
Capacity Increment	`0`
Login Delay Seconds	`0`
Refresh Period	`0`
Allow Shrinking	non-selected

   d. Click Apply.

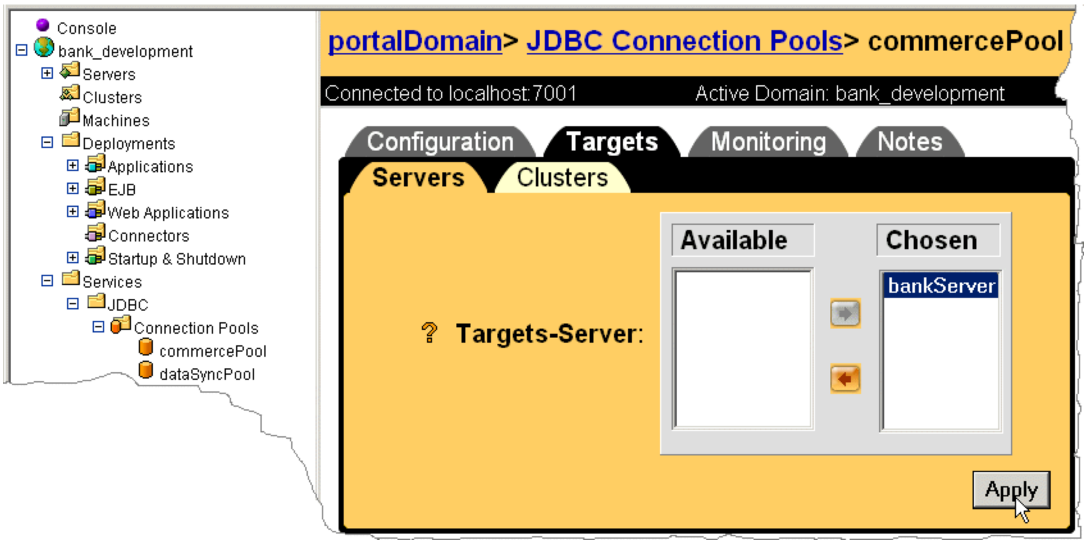
e. Click the Testing tab and enter the following values:

In this box...	**Enter this value...**
Test Table Name	`WEBLOGIC_IS_ALIVE`
Test Reserved Connections	`non-selected`
Test Released Connections	`non-selected`

f. Click Apply.

4. To deploy the commercePool, click the Targets tab. Then move your server from the Available to the Chosen list. (See Figure 3-6.)

Figure 3-6 Deploy commercePool



Tip: Cloning Resources

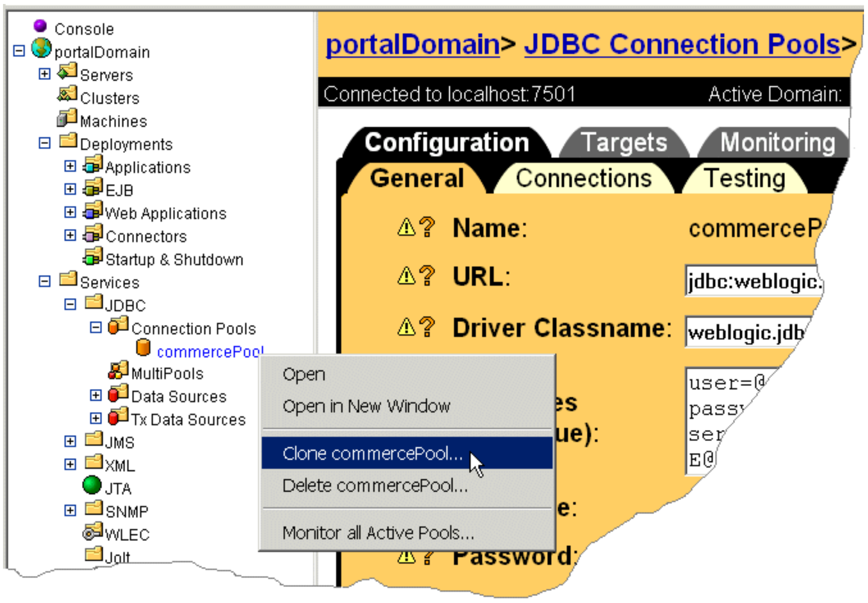
If you are creating a new server, you can clone the commercePool properties to simplify the creation of dataSyncPool.

After you create commercePool, do the following:

1. In the WebLogic Server Administration Console, in the left pane, right click commercePool and select Clone commercePool. (See Figure 3-7.)

   The WebLogic Server Administration Console creates a new connection pool with properties identical to commercePool.

Figure 3-7 Clone CommercePool



Name: commerceP
URL: jdbc:weblogic.
Driver Classname: weblogic.jdb

2. On the Clone page, change the value in the Name box to dataSyncPool.

3. Click Clone.

Configure dataSyncPool for Cloudscape Databases

Create a JDBC connection pool named `dataSyncPool` for Cloudscape by doing the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Right click the Connection Pools folder and click Configure a new JDBC Connection Pool. On the Configure a new JDBC Connection Pool page click the General tab. In the Name box, enter `dataSyncPool`.
3. To configure dataSyncPool, do the following:
   a. On the General tab, enter the following values:

In this box...	**Enter this value...**
URL	`jdbc:cloudscape:Commerce;create=true;upgrade=true`
Driver Classname	`COM.cloudscape.core.JDBCDriver`

Properties	`user=none`
	`password=none`
	`weblogic.t3.waitForConnection=true`
	`weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.waitSecondsForConnectionSecs=999999999999,verbose=false`
	`server=none`

   b. Click Create.

c. Click the Connections tab and enter the following values:

In this box...	Enter this value...
Initial Capacity	`1`
Maximum Capacity	`5`
Capacity Increment	`1`
Login Delay Seconds	`0`
Refresh Period	0
Allow Shrinking	non-selected

d. Click Apply.

e. Click the Testing tab and enter the following values:

In this box...	Enter this value...
Test Table Name	`WEBLOGIC_IS_ALIVE`
Test Reserved Connections	`non-selected`
Test Released Connections	`non-selected`

f. Click Apply.

4. To deploy the dataSyncPool, click the Targets tab. Then move your server from the Available to the Chosen list.

Configure JDBC Data Sources

The reference servers already configure the JDBC data sources that you need to access any database type. If you are modifying a reference server to meet your specific business needs, you do not need to change the configuration for the JDBC data sources.

If you are creating a new server, you must set up JDBC data sources by doing the following:

1. In the left pane of the WebLogic Server Administration Console, open the JDBC folder.
2. Create a **non-transactional commercePool** data source by doing the following:
   a. Right click the Data Sources folder and click Configure a new JDBC Data Source.
   b. On the Configure a new JDBC Data Source page click the Configuration tab and enter the following values:

In this box...	Enter this value...
Name	`commercePool`
JNDI Name	`weblogic.jdbc.pool.commercePool`
Pool Name	`commercePool`
Row Prefetch Enabled	`non-selected`
Stream Chunk Size	256

   c. Click Create.
3. To deploy the data source, click the Targets tab. Then move your server from the Available to the Chosen list. Then click Apply.

4. Create a **transactional commercePool** data source by doing the following:

   a. Right click the **Tx Data Sources** folder and click Configure a new JDBC Tx Data Source.

   b. On the Configure a new JDBC Tx Data Source page click the Configuration tab and enter the following values:

In this box...	**Enter this value...**
Name	`commercePool`
JNDI Name	`weblogic.jdbc.jts.commercePool`
Pool Name	`commercePool`
Enable Two-Phase Commit	`non-selected`
Row Prefetch Enabled	non-selected
Stream Chunk Size	256

   c. Click Create.

5. To deploy the data source, click the Targets tab. Then move your server from the Available to the Chosen list. Then click Apply.

6. Create a **transactional dataSyncPool** data source by doing the following:

   a. Right click the **Tx Data Sources** folder and click Configure a new JDBC Tx Data Source.

b. On the Configure a new JDBC Tx Data Source page click the Configuration tab and enter the following values:

In this box...	Enter this value...
Name	`dataSyncPool`
JNDI Name	`weblogic.jdbc.jts.dataSyncPool`
Pool Name	`dataSyncPool`
Enable Two-Phase Commit	`non-selected`
Row Prefetch Enabled	non-selected
Stream Chunk Size	256

c. Click Create.

7. To deploy the data source, click the Targets tab. Then move your server from the Available to the Chosen list. Then click Apply.

Viewing the JNDI Tree

From time to time you may find it useful to view the objects in the WebLogic Portal JNDI tree. To view the JNDI tree for a server:

1. Right-click the server node in the left pane. This displays a pop-up menu.
2. Select View JNDI Tree. The JNDI tree for this server displays in the right pane.

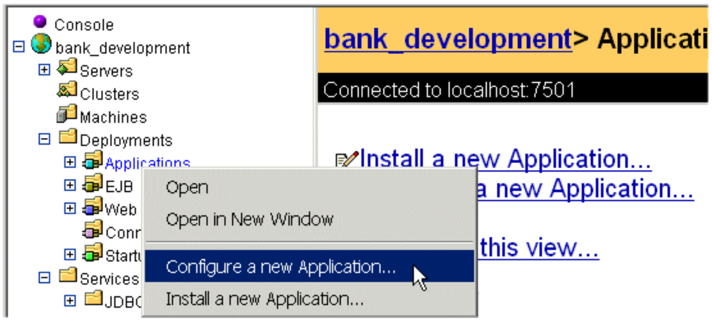
Add Extensions to the WebLogic Server Administration Console

WebLogic Portal includes extensions to the WebLogic Server Administration Console. You use these extensions to configure WebLogic Portal services.

To add the extensions to the WebLogic Server Administration Console, do the following:

1. Make sure that the application p13nConsoleApp is on the computer that hosts the application server. By default, WebLogic Portal installs this application as `PORTAL_HOME/applications/p13nConsoleApp`.
2. Launch the WebLogic Server Administration Console.
3. In the left pane, open the Deployments folder.
4. Right click the Applications folder and click Configure a New Application. (See Figure 3-8.)

Figure 3-8 Configure a New Application



5. On the Configure a New Application page, do the following:

   a. In the Name box, enter p13nConsoleApp. WebLogic Server uses this name only within the WebLogic Server Administration Console. The name does not affect the URIs of resources within the enterprise application.

   b. In the Path box, enter the pathname of the application's root directory. By default, WebLogic Portal installs this application as `PORTAL_HOME/applications/p13nConsoleApp`. To use the default configuration, enter `./applications/p13nConsoleApp`

   c. Select the Deploy check box.

   d. Click Create.

6. In the left pane, open the Servers folder and click the server on which you want to deploy the extensions.

7. On the Server Page, click the Deployments tab. Then click the Web Applications subtab.

8. On the Web Applications subtab, move the p13nConsole Web application from the Available to the Selected list. Then click Apply.

   When you deploy a WebLogic Portal application that uses MBeans to configure its services, the application will now include a Service Configuration folder. This folder contains the WebLogic Portal extensions to the WebLogic Server Administration Console. (See Figure 3-9.)

Note: If your enterprise application does not contain the WLS-specific configuration file, `WEB-INF/application-config.xml`, then you must create one by stopping your server and taking the following additional steps.

9. Create an `application-config.xml` file under:
   `<portal40home>/applications/<YourEnterpriseApp>/META-INF/`

   You can use the existing Portal enterprise applications as models by copying one of those files. Make sure the services that you want to be able to configure are present. For example, for the JDBC Helper Service:

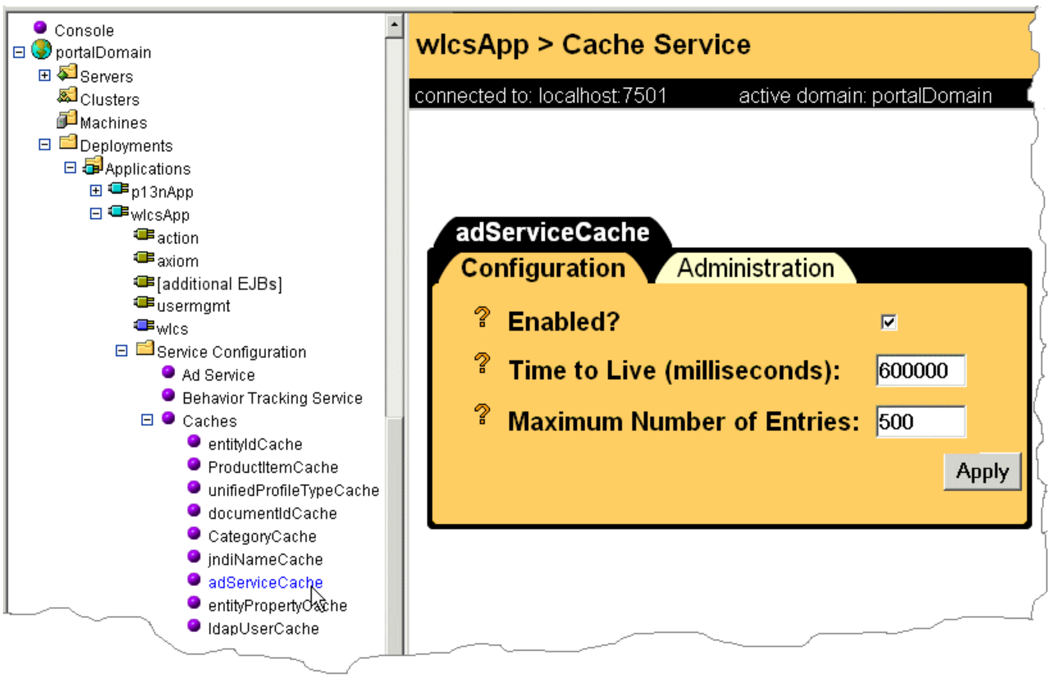
```
<ApplicationConfiguration>
  <JdbcHelper
    JdbcHelperDelegate=""
    MaxRetries="-1"
    MaxWaitTime="-1"
    Name="JdbcHelper"
  />
</ApplicationConfiguration>
```

10. To register your `application-config.xml` file with your enterprise application, edit `config.xml` for your domain by adding the `<ApplicationConfiguration>` tag to the body of the `<Application>` tag for your enterprise application:

```
    <Application Deployed="true" Name="petflowApp"
Path=".\applications\petflowApp">
        <WebAppComponent IndexDirectoryEnabled="false"
Name="petflow"
        Targets="petflowServer" URI="petflow"/>
        <ApplicationConfiguration
        Name="petflowApp"
        Targets="petflowServer"
        URI="META-INF/application-config.xml" />
        <EJBComponent Name="pipeline" Targets="petflowServer"
URI="pipeline.jar"/>
        <EJBComponent Name="petflow" Targets="petflowServer"
URI="petflow.jar"/>
        <EJBComponent Name="petStore_EJB" Targets="petflowServer"
URI="petStore_EJB.jar"/>
        <WebAppComponent IndexDirectoryEnabled="false"
Name="datasync"
        Targets="petflowServer" URI="datasync"/>
    </Application>
```

11. Restart your server.

Figure 3-9 Extensions to the WebLogic Server Administration Console



connected to: localhost:7501 active domain: portalDomain

Enabled? ✓

Time to Live (milliseconds): 600000

Maximum Number of Entries: 500

Make weblogiccommerce.properties Available to the Domain

The `PORTAL_HOME/weblogiccommerce.properties` is an additional file that configures WebLogic Portal services. It is deprecated along with the sections of Java code that it configures.

Specify a Default Web Application

Every server and virtual host in your domain can declare a default Web Application. The default Web application responds to any HTTP request that cannot be resolved to another deployed Web application. In contrast to all other Web applications, the default Web application does not use the Web application name as part of the URI. Any Web application targeted to a server or virtual host can be declared as the default Web application.

For information on how to specify a default Web application, refer to “Configuring WebLogic Server Web Components” in the *WebLogic Server Administration Guide*.

Relocate the Domain (Optional)

If you want to relocate the domain into a different directory tree, perhaps on a different disk partition or in a directory tree that is separate from `portalDomain`, do the following:

1. Create a new domain repository by creating a directory named `config`.
2. Copy your domain directory into the new `config` directory.

For example, on a partition named `usr2`, you create the following tree:

```
/usr2/config/bank_development
/usr2/config/bank_development/config.xml
/usr2/config/bank_development/SerializedSystemIni.dat
```

If you locate your `config` directory and your domain’s startup script in different directories, add the following argument to the Java command that is near the end of the script:

```
-Dweblogic.RootDirectory=path-to-the-parent-directory-of-your-conf
ig-directory
```

For example, you want to locate the startup script in `/usr/local/bin` and the `config` directory in `/usr2`. Modify the startup script by adding the bold text in the following example:

```
REM ----- Start WebLogic with the above parameters ------

%JDK_HOME%\bin\java %JAVA_VM% -Xms128m -Xmx128m -classpath
%CLASSPATH% -Dcloudscape.system.home=%PORTAL_HOME%/db/data
-Dweblogic.Domain=%DOMAIN_NAME% -Dweblogic.Name=%SERVER_NAME%
-Dbea.home=%BEA_HOME%
-Djava.security.policy==%WEBLOGIC_HOME%/lib/weblogic.policy
-Dcommerce.properties=%PORTAL_HOME%/weblogiccommerce.properties
-Dpipeline.properties=%PORTAL_HOME%/pipeline.properties
-Dwebflow.properties=%PORTAL_HOME%/webflow.properties
-Dweblogic.RootDirectory=/usr2
weblogic.Server
```

Add the `-Dweblogic.RootDirectory` argument in front of `weblogic.Server`.

Place the New Directory Tree Under Source Control

To safeguard your work, keep a history of changes, and provide the ability to distribute a known version of source files to your development team, place your new domain and server configuration under source control. Do not place the generated directories, such as `/config/applications/.wlnotdelete`, under source control.

For information on changing other properties in the WebLogic Server Administration Console, refer to the following documents:

- *Performance Tuning Guide*
- *BEA WebLogic Server Administration Guide*
- *Administration Console Online Help*

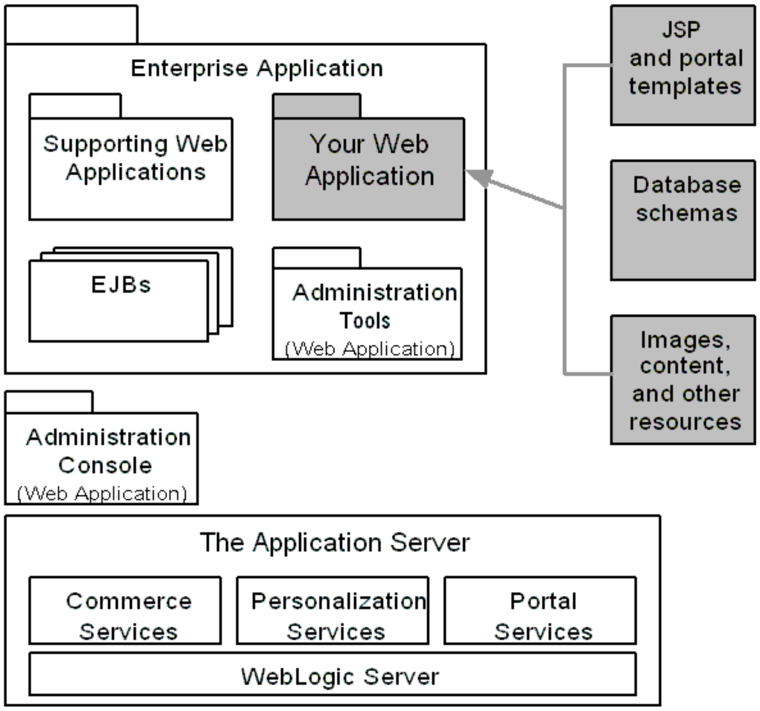
CHAPTER

4 Assembling Your Web Application

On your e-business Web site, you use a Web application to implement your business model. It is one part (module) of an overall enterprise application that you assemble and deploy. (See Figure 4-1.)

With this modular approach, your development efforts focus on implementing your business model, and BEA provides the remaining infrastructure and tools for deploying and managing your Web application.

Figure 4-1 Your Web Application in Context



This topic describes how to assemble a Web application on a WebLogic Portal application server. It assumes that, at the very least, you have designed a prototype of your Web application and created skeletal JSPs. For more information about designing a prototype, refer to "Milestone 1" under "Workflow for Developing an E-Business Web Site" in the *Strategies for Developing E-Business Web Sites*.

This topic includes the following sections:

- Create and Populate a Directory Tree
- Create Deployment Descriptors

The next topic, "Assembling and Deploying Enterprise Applications" on page 5-1, describes how to deploy your Web application within an enterprise application.

Create and Populate a Directory Tree

You develop your Web application within a specified directory structure so that it can be archived and deployed on the application server. All servlets, classes, static files, and other resources belonging to a Web application are organized under a directory hierarchy. The root of this hierarchy defines the document root of your Web application. All files under this root directory can be served to the client, except for files under the special directory `WEB-INF`, located under the root directory. The name of the root directory is used to resolve requests for components of the Web application.

For more information about the directory structure of Web applications, refer to *WebLogic Server Assembling and Configuring Web Applications* guide.

For information specific to developing a WebLogic Portal Web application, refer to the following sections:

- Using a Reference Web Application as a Template
- Add JSP Tag Libraries to WEB-INF

Using a Reference Web Application as a Template

WebLogic Portal installs Web applications within the reference enterprise applications. If one of the samples approximates your business needs, you can duplicate and modify the sample. If you modify a sample Web application, make sure that you do the following before you deploy the Web application:

- Rename the root directory of your copy to reflect the purpose of your Web application.
- In the `WEB-INF` directory of your copy, remove any directories with names that start with `_tmp`. These directories contained compiled versions of JSPs that you have visited.

- In the root directory, `index.jsp` redirects requests to a Webflow for your Web application. The Webflow then determines which JSP (or portal) to display based on the Webflow properties that you create. The redirect statement follows this syntax:
  `<jsp:forward page="/application?namespace=`*`namespace_name`*`">`
  where *`namespace_name`* is the initial namespace for your Web application's Webflow.

  Change the *`namespace_name`* component of this statement to refer to the initial Webflow namespace of your Web application. For more information, refer to "Overview of Webflow" in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Add JSP Tag Libraries to WEB-INF

WebLogic Portal includes several JSP tag libraries which support the JSPs tags in your Web application. The tag libraries reside in JAR files that you must copy to your Web application. Each JAR file is named with the following convention: *`service`*`_taglib.jar`.

Copy each tag-library JAR file that your Web application uses to the `WEB-INF/lib` directory in your Web application. The reference copies are located in the following directories:

- `PORTAL_HOME/lib/p13n/web`
- `PORTAL_HOME/lib/commerce/web`
- `PORTAL_HOME/lib/portal/web`
- `BEA_HOME/wlserver6.1/ext`. The WebLogic Server tag libraries do not use the same naming convention as WebLogic Portal. In most cases, you need to copy only the `weblogic-tags.jar` file into your Web application.

For each tag library that you copy, you must also declare it in the Web application's `web.xml` file. For more information, refer to "Taglib Declarations" on page 4-17.

Create Deployment Descriptors

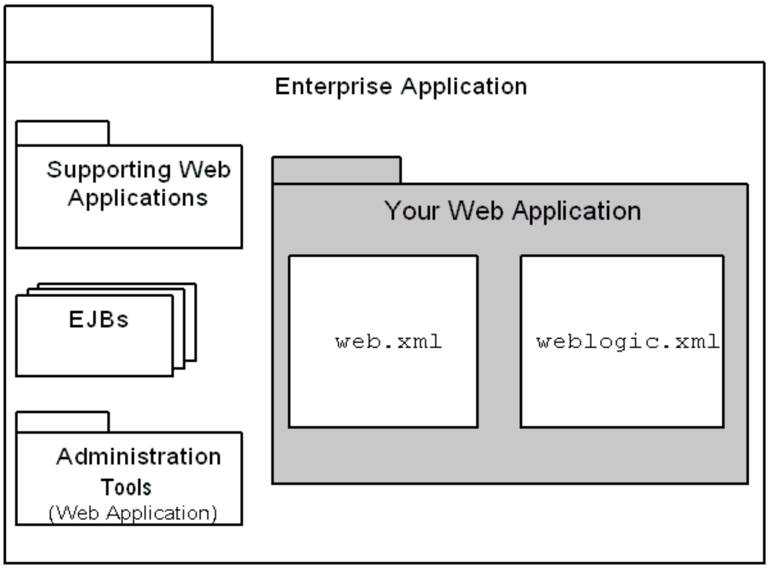
Each Web application uses two deployment descriptors: `web.xml`, which contains standard J2EE properties and `weblogic.xml`, which contains properties that are specific to WebLogic Portal.

To create deployment descriptors for your Web application, do the following:

- Copy Reference Files
- Review and Modify web.xml
- Modify weblogic.xml

The `web.xml` and `weblogic.xml` files contain properties that are scoped to a single Web application; they do not configure the parent enterprise application or any other modules in the enterprise application.

Figure 4-2 Scope of web.xml and weblogic.xml


Enterprise Application
Supporting Web Applications
EJBs
Administration Tools (Web Application)
Your Web Application
web.xml
weblogic.xml

Copy Reference Files

The reference deployment descriptors declare servlets, security roles and secured resources, JSP deployment options and other properties for a Web application.

To copy these files to your Web applications, do the following:

1. In the root directory of your Web application, create a directory named `WEB-INF`. For example:

   `myWebApplication``/WEB-INF`

2. Find a reference Web application that approximates your business needs. Then copy its `web.xml` and `weblogic.xml` files to your `WEB-INF` directory. For example, to copy files from the wlcs Web application, copy the following files into your `WEB-INF` directory:

```
PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF/web.xml
PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF/weblogic.xml
```

Review and Modify web.xml

Open your `web.xml` file in a text editor to review its properties. You might need to modify some properties to support your environment.

Note: This document describes using a text editor to create and modify deployment descriptors. If you have deployed your Web application onto an active server, you can also use the WebLogic Server Administration Console to edit the deployment descriptors.

The elements in your `web.xml` file must appear in the following order:

- Display Name and Description
- Webflow Default Namespace
- URLs for the Webflow Service
- Listen Ports for Webflow-Generated URLs
- Declarations of Secure Links
- Filter Objects
- Listener Objects
- Servlet Registration and Mappings
- Session Configuration
- Welcome and Error Pages
- Taglib Declarations
- Name for commercePool Data Source
- Declarations of Secure JSPs
- Login Configuration
- Declaration of Security Roles
- EJB References

For a general description of properties in `web.xml` files, refer to “web.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* guide.

Display Name and Description

The `<display-name>` element determines the name that the WebLogic Server Administration Console uses for your Web application. Customers do not see this name.

The `<description>` element is optional. WebLogic Server Administration Console does not display the value of this element.

For more information, refer to "web.xml Deployment Descriptor Elements" in the *WebLogic Server Assembling and Configuring Web Applications* guide.

Webflow Default Namespace

Although you will generally have only one Webflow per Web application, **namespaces** can be used to separate a Webflow into a number of smaller, more manageable modules. Namespaces allow multiple developers to work with separate portions of a Webflow (and thus with separate configuration files and Pipeline Session properties) that feed into the larger Webflow for a Web application, without having to worry about naming collisions.

The following property declares the default namespace for the Web application:

```
<context-param>
      <param-name>P13N_DEFAULT_NAMESPACE</param-name>
      <param-value>namespace</param-value>
</context-param>
```

Make sure that the `<param-value>` matches the name of the Web application's default namespace. You use the E-Business Control Center to create Webflow namespaces. For more information, refer to "Webflow Components" in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

URLs for the Webflow Service

Several elements in `web.xml` configure the URLs that Webflow generates:

- The following element determines the URL that Webflow uses to request the WebLogic Portal Administration Tools:

  ```
  <context-param>
        <param-name>WLCS_ADMIN_URL</param-name>
        <param-value>/tools/application/admin</param-value>
  </context-param>
  ```

  The wlcs sample application uses this element because it provides a link to the WebLogic Portal Administration Tools in the heading of its sample JSPs. In most environments, you do not need to include this element.

- The following element inserts `/weblogic` as a prefix for URLs that Webflow generates:

  ```
  <context-param>
        <param-name>P13N_URL_PREFIX</param-name>
        <param-value>/weblogic</param-value>
  </context-param>
  ```

  Include this element if you use a third party Web server (such as Apache) or other service in front of WebLogic Server and WebLogic Portal.

- The following element specifies a context for URLs that point to static resources that exist on a proxy server:

  ```
  <context-param>
        <param-name>P13N_STATIC_ROOT</param-name>
        <param-value></param-value>
     </context-param>
  ```

  Use this element in conjunction with the `createStaticResourceURL()` method to create links to static resources.

Listen Ports for Webflow-Generated URLs

The elements in this section enable Webflow to encode port numbers in URLs that are different from the port numbers that the WebLogic Server uses. For example, use these elements if you want Webflow-generated URLs to specify the port numbers of a proxy server.

```
<context-param>
      <param-name>HTTP_PORT</param-name>
      <param-value>port-number</param-value>
</context-param>

<context-param>
      <param-name>HTTPS_PORT</param-name>
      <param-value>port-number</param-value>
</context-param>

<context-param>
       <param-name>P13N_URL_DOMAIN</param-name>
       <param-value>host-name</param-value>
</context-param>
```

If you want the URLs in your Web application to use the same port numbers as the WebLogic Server instance, either deactivate these elements by surrounding them in comment tags,

<!-- -->, or change the values to match the listen port properties for the server.

Declarations of Secure Links

The following element syntax declares a set of files, pipelines, and input processors that need to be accessed via HTTPS. When the `createWebflowURL()` method encounters one of the resources you declare in the `<param-value>` subelement, it generates a URL that uses the HTTPS protocol.

```
<context-param>
   <param-name>HTTPS_URL_PATTERNS</param-name>
   <param-value>URLs-and-URL patterns</param-value>
</context-param>
```

You can add specific resource names or name patterns to the `<param-value>` element. All *`URLs-and-URL patterns`* must start with the name of a Webflow namespace. For example, the following line specifies that `secureMain.jsp` is a secured resource when called from the `sampleapp_main` Webflow namespace:
`/sampleapp_main/secureMain.jsp`

If you add any target names or patterns to the `<param-value>` element, you must also add them to the `<security-constraint>` element, which is described in "Declarations of Secure JSPs" on page 4-18.

Filter Objects

If you use the Event and Behavior Tracking services, include the following filter elements to forward ad clickthrough requests from Webflow to the Events Service:

```
<filter>
    <filter-name>ClickThroughEventFilter </filter-name>

    <filter-class>
    com.bea.p13n.tracking.clickthrough.ClickThroughEventFilter
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>ClickThroughEventFilter </filter-name>
     <servlet-name>webflow</servlet-name>
</filter-mapping>
```

Listener Objects

If your application uses personalization or commerce services, you must register the `AnonymousProfileListener`, which listens for the creation of user sessions. When the server creates a user session, `AnonymousProfileListener` initializes the Anonymous user profile. Your application stores data in the Anonymous user profile until the customer logs in. If you want to maintain session information for anonymous customers, include the following elements in web.xml:

```
<listener>
  <listener-class>
  com.bea.p13n.servlets.AnonymousProfileListener
  </listener-class>
</listener>
```

If you use the Event and Behavior Tracking services, include the following elements to notify the services of when the server begins and ends sessions:

```
<listener>
  <listener-class>
  com.bea.p13n.tracking.listeners.SessionEventListener
  </listener-class>
</listener>
```

Servlet Registration and Mappings

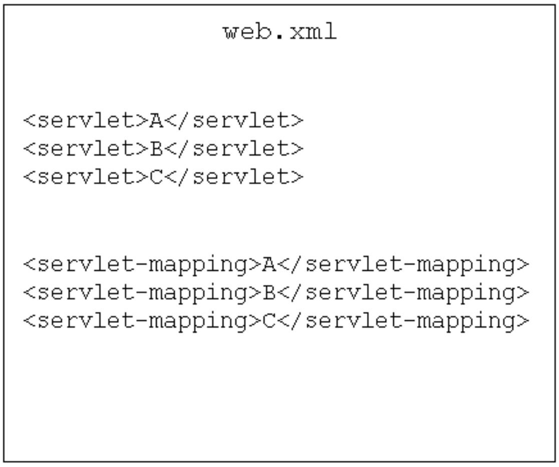
The reference Web applications register the following servlets and define mappings between the servlet and a URL pattern:

- Servlets for Webflow Services
- Servlets for Personalization Services
- Servlets for Commerce Services

You can remove a declaration and mapping if your Web application does not use the servlet. If you create your own servlet to be used by this Web application, you must register it in the `web.xml` file. For information about the `<servlet>` and `<servlet-mapping>` elements, refer to "web.xml Deployment Descriptor Elements" in the *WebLogic Server Assembling and Configuring Web Applications* guide.

Place all servlet declarations in one group and all servlet mappings into a separate group. Place the declarations group immediately before the mappings group. (See Figure 4-3.)

Figure 4-3   Group Declarations and Mappings



```
                web.xml


<servlet>A</servlet>
<servlet>B</servlet>
<servlet>C</servlet>


<servlet-mapping>A</servlet-mapping>
<servlet-mapping>B</servlet-mapping>
<servlet-mapping>C</servlet-mapping>
```

Servlets for Webflow Services

Include the XML element from Table 4-1 to declare servlets and map URL patterns for Webflow services. Note that all `<servlet>` elements must be grouped together and all `<servlet-mapping>` must be grouped together. (See Figure 4-3.)

Table 4-1 Servlet Declarations for Webflow Services

Servlet	Declaration and Mapping

`Webflow`
for Web applications that **do not use** portal services
	`<servlet>`
	`  <servlet-name>webflow</servlet-name>`
	`  <servlet-class>`
	`   com.bea.p13n.appflow.webflow.servlets.internal.WebflowServlet`
	`  </servlet-class>`
	`</servlet>`
	`...`
	`<servlet-mapping>`
	`  <servlet-name>webflow</servlet-name>`
	`  <url-pattern>/application/*</url-pattern>`
	`</servlet-mapping>`

`Webflow`
for Web applications that use portal services
	`<servlet>`
	`  <servlet-name>webflow</servlet-name>`
	`  <servlet-class>`
	`   com.bea.portal.appflow.servlets.internal.PortalWebflowServlet`
	`  </servlet-class>`
	`</servlet>`
	`...`
	`<servlet-mapping>`
	`  <servlet-name>webflow</servlet-name>`
	`  <url-pattern>/application/*</url-pattern>`
	`</servlet-mapping>`

Servlets for Personalization Services

Include the XML elements from Table 4-2 to declare servlets and map URL patterns for personalization services. Note that all `<servlet>` elements must be grouped together and all `<servlet-mapping>` must be grouped together. (See Figure 4-3.)

Table 4-2 Servlet Declarations for Personalization Services

Servlet	Declaration and Mapping
ShowDocServlet Outputs the contents of a `Document` object. This servlet is useful when streaming the contents of an image that resides in a content management system. For more information, refer to “Creating and Managing Content” in the *Guide to Building Personalized Applications*.	`<servlet> <servlet-class> com.bea.p13n.content.servlets.ShowDocServlet </servlet-class> <!-- Make showdoc always use the local ejb-ref DocumentManager --> <init-param> <param-name>contentHome</param-name> <param-value> java:comp/env/ejb/DocumentManager </param-value> </init-param> </servlet> ... <servlet-mapping> <servlet-name>ShowDocServlet</servlet-name> <url-pattern>/ShowDoc/*</url-pattern> </servlet-mapping>`
`clickThrough` Redirects a user to an external site and fires a click event.	`<servlet> <servlet-name>clickThroughServlet</servlet-name> <servlet-class> com.bea.p13n.tracking.clickthrough.ClickThroughServlet </servlet-class> </servlet> ... <servlet-mapping> <servlet-name>clickThroughServlet</servlet-name> <url-pattern>/clickThroughServlet/*</url-pattern> </servlet-mapping>`

Table 4-2 Servlet Declarations for Personalization Services (Continued)

Servlet	Declaration and Mapping

`adClickThru`
Supports ad placeholders by recording click events and redirecting the browser to the ad's click-thru location.
For more information, refer to "Working with Ad Placeholders" in the *Guide to Building Personalized Applications*.
	`<servlet>`
	`  <servlet-name>adClickThru</servlet-name>`
	`  <servlet-class>`
	`   com.bea.p13n.ad.servlets.AdClickThruServlet`
	`  </servlet-class>`
	`</servlet>`
	`...`
	`<servlet-mapping>`
	`  <servlet-name>adClickThru</servlet-name>`
	`  <url-pattern>/AdClickThru/*</url-pattern>`
	`</servlet-mapping>`

Servlets for Commerce Services

Include the XML elements from Table 4-3 to declare servlets for commerce services. You do not need to map these servlets to support WebLogic Portal services.

Table 4-3 Servlet Declarations for Commerce Services

Servlet	Declaration

`Security`
Used only by site administrators to specify encryption values for securing credit card transactions.
	`<servlet>`
	`    <servlet-name>Security</servlet-name>`
	`    <servlet-class>`
	`com.beasys.commerce.ebusiness.security.EncryptionServlet`
	`    </servlet-class>`
	`</servlet>`
	`<servlet>`
	`    <servlet-name>KeyGenerator</servlet-name>`
	`    <servlet-class>`
	`com.beasys.commerce.ebusiness.security.KeyGeneratorServlet`
	`    </servlet-class>`
	`</servlet>`

Session Configuration

The `<session-config>` element defines the session parameters for this Web Application. The WebLogic Portal sample applications use the `<session-timeout>` parameter to determine how many minutes of inactivity the server will tolerate before ending the session. Configure this to a suitable interval for your environment.

For more information, refer to "web.xml Deployment Descriptor Elements" in the *WebLogic Server Assembling and Configuring Web Applications* guide.

Welcome and Error Pages

The `<welcome-file-list>` element specifies a main page for your web application.

The `<error-page>` elements map error codes that the HTTP server returns to HTML files or JSPs.

For more information, refer to "web.xml Deployment Descriptor Elements" in the *WebLogic Server Assembling and Configuring Web Applications* guide.

Taglib Declarations

WebLogic Portal provides several libraries of JSP tags. You use the tags to implement WebLogic Portal features such as Webflow, Pipelines, and placeholders.

You declare each tag library that you use in your Web application. Use the following syntax to declare a tag library:

```
<taglib>
    <taglib-uri>library.tld </taglib-uri>
    <taglib-location>pname-to-JAR-file</taglib-location>
</taglib>
```

The pathname for the JAR file that contains the library is relative to the root of the Web application. For example,

```
<taglib>
   <taglib-uri> weblogic.tld </taglib-uri>
   <taglib-location>/WEB-INF/lib/weblogic-tags.jar
   </taglib-location>
</taglib>
```

Name for commercePool Data Source

The following element declares a name for the `commercePool` data source:

```
<resource-ref>
   <res-ref-name>jdbc/commercePool</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
    <res-auth>CONTAINER</res-auth>
 </resource-ref>
```

The `weblogic.xml` file maps the `commercePool` data source to a JNDI name, and the domain configures the data source. For information on the data source declaration in `weblogic.xml`, refer to “Mapping of commercePool Data Source” on page 4-22. For information on defining a data source for the domain, refer to “Set Up JDBC Connection Pools and Data Sources” on page 3-17.

Declarations of Secure JSPs

The `<security-constraint>` element declares a collection of resources (JSPs) that only specific roles can access. For more information, refer to “web.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* guide.

The example in Listing 4-1 is from the wlcs Web application. It secures all JSPs that are under the `wlcs/user` and `wlcs/order` directories and specifies that only users who are authenticated for `CustomerRole` can access those JSPs.

To give another role access to the resource collection in Listing 4-1, add a `<role-name>` element to the `<auth-constraint>` element. If you add a role name, you must add a declaration for the role name as described in “Declaration of Security Roles” on page 4-20.

To secure additional directories or specific files, add `<url-pattern>` elements. Note that a pattern or filename must start with a `/` character (forward slash). If you add any `<url-pattern>` elements, you must also add those patterns to the patterns defined for `<param-name>HTTPS_URL_PATTERNS</param-name>`. For more information, see “Declarations of Secure Links” on page 4-11.

Listing 4-1 Security Constraints

```
<security-constraint>
    <!-- Define a resource collection -->
    <web-resource-collection>
         <web-resource-name>Customer Profile - Self Administration Pages
         </web-resource-name>
         <description>Customer Profile - Self Administration Pages</description>

         <!-- URL pattern for the resource collection -->
         <url-pattern>/commerce/user/*</url-pattern>
         <url-pattern>/commerce/order/*</url-pattern>

         <http-method>GET</http-method>
         <http-method>POST</http-method>

    </web-resource-collection>

  <!-- This constraint applies to users with role "CustomerRole" -->

      <auth-constraint>
         <description>Users with role "CustomerRole"</description>
         <role-name>CustomerRole</role-name>
      </auth-constraint>

      <!-- For enabling SSL, specify CONFIDENTIAL or INTEGRAL. -->

      <user-data-constraint>
         <transport-guarantee>CONFIDENTIAL</transport-guarantee>
      </user-data-constraint>

</security-constraint>
```

Login Configuration

The `<login-config>` element determines how users are authenticated. All of the WebLogic Portal sample applications use an HTML form (`<auth-method>FORM</auth-method>`) to authenticate users. If you use the `FORM` method, you can also use the `P13NAuthFilter` to initialize customer profiles when a customer logs on. For more information, refer to “Customer Profile Initialization” on page 4-26.

Declaration of Security Roles

You must include a `<security-role>` element to declare all of the roles that you use in the `<security-constraint>` (described in "Declarations of Secure JSPs" on page 4-18).

For example, the wlcs sample application gives `CustomerRole` access to secured JSPs (see Listing 4-1 above). Therefore, the wlcs `web.xml` file includes the following element:

```
<security-role>
  <description>Registered customers with role CustomerRole
  </description>
  <role-name>CustomerRole</role-name>
</security-role>
```

In addition, the wlcs `weblogic.xml` file includes a `<security-role-assignment>` element, which maps the `CustomerRole` to a group name (principal) from the wlcsRealm. For more information about mapping security roles to principles, refer to "Security-Role Mappings" on page 4-22.

EJB References

The `web.xml` file must specify the home and remote interfaces for each EJB that the Web application uses. For example, the following XML element specifies interfaces for the `AdService` EJB:

```
<ejb-ref>
     <description> The AdService for this webapp</description>
     <ejb-ref-name>ejb/AdService</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <home>com.bea.p13n.ad.AdServiceHome</home>
     <remote>com.bea.p13n.ad.AdService</remote>
</ejb-ref>
```

For information on including these XML elements in your `web.xml` file, refer to "Add JAR Files" on page 5-27.

Modify weblogic.xml

In addition to the `web.xml` deployment descriptor, each WebLogic Portal Web application uses `weblogic.xml` to declare deployment properties that are specific to WebLogic Server.

After you copy a reference `weblogic.xml` to your Web application, review or modify the following properties:

- Security-Role Mappings
- Mapping of commercePool Data Source
- Check Authentication on JSP Forwarding
- JNDI Names for EJBs
- Cookies
- Support for Clusters
- JSP Deployment Options
- Customer Profile Initialization

For general information about `weblogic.xml`, refer to “weblogic.xml Deployment Descriptor Elements” in *WebLogic Server Assembling and Configuring Web Applications*.

Security-Role Mappings

Each Web application uses the following element syntax to map a J2EE security role to a user or group that a security realm uses to define ACLs:

```
<security-role-assignment>
      <role-name>J2EE-role</role-name>
     <principal-name>security-realm user or group</principal-name>
</security-role-assignment>
```

For example, WebLogic Portal uses the following element to map settings that its security realm defines for `wlcs_customer` to settings that `application.xml` defines for the `CustomerRole` J2EE security role:

```
<security-role-assignment>
   <role-name>CustomerRole</role-name>
   <principal-name>wlcs_customer</principal-name>
</security-role-assignment>
```

If you add groups to your security realm, you must add mappings to this section of `weblogic.xml`.

Mapping of commercePool Data Source

The following element maps the `commercePool` data source to a JNDI name:

```
<resource-description>
   <res-ref-name>jdbc/commercePool</res-ref-name>
   <jndi-name>weblogic.jdbc.jts.commercePool</jndi-name>
</resource-description>
```

The `web.xml` file declares a name for the commercePool data source, and the domain configures the data source. For information about the declaration in the `web.xml` file, refer to “Name for commercePool Data Source” on page 4-18. For information on defining a data source for the domain, refer to “Set Up JDBC Connection Pools and Data Sources” on page 3-17.

Check Authentication on JSP Forwarding

The following element requires authentication of any forwarded requests from a servlet or JSP:

```
<container-descriptor>
   <check-auth-on-forward/>
</container-descriptor>
```

The Webflow service requires this element. For more information, refer to "weblogic.xml Deployment Descriptor Elements" in the *WebLogic Server Assembling and Configuring Web Applications* guide.

JNDI Names for EJBs

The `weblogic.xml` file must specify the JNDI names for each EJB that the Web application uses. Each declaration must be within the `<reference-descriptor>` element.

For example, the following XML subelement specifies the JNDI name for the `AdService` EJB:

```
<reference-descriptor>

  <ejb-reference-description>
     <ejb-ref-name>ejb/AdService</ejb-ref-name>
     <jndi-name>
      ${APPNAME}.BEA_personalization.AdService
     </jndi-name>
  </ejb-reference-description>
```

... additional `<ejb-reference-description>` elements ...

```
</reference-descriptor>
```

For information on including these XML elements in your `weblogic.xml` file, refer to "Add JAR Files" on page 5-27.

Support for Clusters

If you deploy your Web application in a cluster, add the following elements to enable replication of HTTP Sessions across the cluster nodes:

```
<session-param>
  <param-name>PersistentStoreType</param-name>
  <param-value>replicated</param-value>
</session-param>
```

For information on other values for this element, refer to “weblogic.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* guide.

For more information about clustering, refer to the following:

- Chapter 6, “Deploying Clusters.”
- *Using WebLogic Server Clusters*, which is part of the BEA WebLogic Server documentation set.

Cookies

WebLogic Server uses cookies for session management when supported by the client browser. The cookies that WebLogic Server uses to track sessions are set as transient by default and do not outlive the life of the browser.

By default, WebLogic Server assigns the same cookie name (`JSESSIONID`) to all Web applications on the current server instance. When you use any type of authentication, all Web applications that use the same cookie name use a single sign-on for authentication. Once a user is authenticated, that authentication is valid for requests to any Web application that uses the same cookie name. The user is not prompted again for authentication.

If you want to require separate authentication for a Web application, you can use the following element to specify a unique cookie name:

```
<session-descriptor>
     <session-param>
        <param-name>CookieName</param-name>
        <param-value>
         unique-cookie-name
        </param-value>
     </session-param>
</session-descriptor>
```

If your Web site supports Netscape 4.5 and higher, include the following element to set the cookie domain for the application.

```
<session-param>
      <param-name>CookieDomain</param-name>
      <param-value>.mycompany.com</param-value>
</session-param>
```

where *`.mycompany.com`* is your domain.

Note that all of the sample WebLogic Portal Web applications specify unique cookie names because they are not designed to be used in conjunction with other Web applications. You can design your WebLogic Portal Web application to be used with other Web applications and to support the WebLogic Server single sign-on feature.

For more information, refer to “Programming Tasks” in the *WebLogic Server Programming WebLogic HTTP Servlets* guide.

JSP Deployment Options

Each Web application includes a `<jsp-descriptor>` element to specify options for deploying JSPs.

For information about the `<jsp-descriptor>` element and its subelements, refer to “weblogic.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* guide.for more info go to WLS doc for this.

For recommended settings for some of the `<jsp-descriptor>` subelements, refer to the *Performance Tuning Guide*

Customer Profile Initialization

All WebLogic Portal Web applications must initialize customer profiles, especially if you want them to use campaigns, Behavior Tracking, and JSP tags that provide personalized content based on data in the customer's profile.

The `P13NAuthFilter` initializes customer profiles when a customer logs in using a form. If you want to use `P13NAuthFilter`, you must use the `FORM` authentication method. For information on configuring your application to use the `FORM` authentication method, refer to "Welcome and Error Pages" on page 4-17.

To use the `P13NAuthFilter`, add the following element to `weblogic.xml`:

```
<auth-filter>com.bea.p13n.servlets.P13NAuthFilter</auth-filter>
```

If you do not want to use `P13NAuthFilter`, you can initialize the customer profile with Webflow components or other methods. For more information, refer to "Setting Up JSP Tags and Scriptlets for Campaigns" in the *Guide to Developing Campaign Infrastructure*.

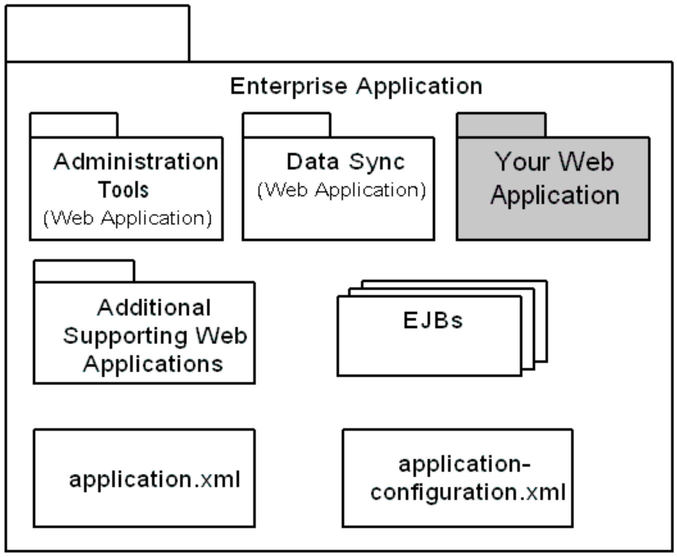
CHAPTER

5 Assembling and Deploying Enterprise Applications

WebLogic Portal requires that you deploy all of your e-business resources within enterprise applications. Each enterprise application contains the following types of components (see Figure 5-1):

- Web applications that deploy data and administer your Web application
- EJBs that provide e-business functions
- Deployment descriptors
- Your Web application

Figure 5-1 Your Enterprise Application


Enterprise Application
Administration Tools (Web Application)
Data Sync (Web Application)
Your Web Application
Additional Supporting Web Applications
EJBs
application.xml
application-configuration.xml

This topic describes how to assemble and deploy an enterprise application on a WebLogic Portal application server:

- Create Deployment Descriptors
- Create a Root Directory
- Add Your Web Application
- Add Supporting Web Applications
- Add JAR Files
- Create an EAR File (Optional)
- Deploy the Application

Create a Root Directory

Create a directory that serves as the root of your application. Choose a directory name that indicates the purpose and scope of the application. The name of the directory is not visible to customers who visit your site.

To support multiple developers in a development environment, you might deploy multiple instances of this enterprise application under different names on a single server. To do so, you must create multiple copies of the application's directory tree and rename the root directories for each instance. For example, you want to deploy multiple instances of an enterprise application whose root directory is name BankApp. You create two copies of the BankApp directory tree and change the root name of one tree to BankApp1 and the other to BankApp2. For more information, refer to "Deploy Multiple Instances of an Application" on page 5-38.

Create Deployment Descriptors

Each enterprise application uses two deployment descriptors: `application.xml`, which contains standard J2EE properties and `application-config.xml`, which contains properties that declare and configure JMX MBeans. WebLogic Portal uses MBeans to dynamically configure application services.

To create deployment descriptors for your enterprise application, do the following:

- Copy Reference Files
- Modify application.xml
- Modify application-config.xml

Copy Reference Files

The reference enterprise applications include deployment descriptors that declare the modules, security roles, and MBeans that are required to support WebLogic Portal.

To copy these files to your enterprise applications, do the following:

1. In the root directory of your application, create a directory named `META-INF`. For example:

   `myApplication``/META-INF`

2. Copy the following files to your `META-INF` directory:

   ```
   PORTAL_HOME/applications/wlcsApp/META-INF/application.xml
   PORTAL_HOME/applications/wlcsApp/META-INF/application-config.xm
   l
   ```

Modify application.xml

Open your `application.xml` in a text editor to make the following modifications:

- Application Name
- Declarations for Web Applications
- Declarations for EJBs
- Declarations of Security Roles

For a general introduction to the elements in `application.xml`, refer to "application.xml Deployment Descriptor Elements" in the *WebLogic Server Developing Applications* guide.

Application Name

At the top of the `application.xml` file is a declaration of the name of the enterprise application. WebLogic Server displays this name in the WebLogic Server Administration Console. Your customers do not see this name.

To provide a unique name for your application, change the values in the following elements:

```
<application>
  <display-name>name-of-your-application</display-name>
  <description>optional-text-description</description>
```

Declarations for Web Applications

In `application.xml` you must declare each Web application that you copy into your enterprise application's directory tree. For information about copying Web applications and declaring them in `application.xml`, refer to the following sections:

- "Add Your Web Application" on page 5-20
- "Add Supporting Web Applications" on page 5-20

Remove any `<web>` modules declaring Web applications that you are not including. Below is an example of a declaration for the `catalogws` Web application:

```
<module>
    <web>
      <web-uri>catalogws-webservice.war</web-uri>
      <context-root>wlcsAppCatalogTool</context-root>
    </web>
</module>
```

Declarations for EJBs

In `application.xml` you must declare each EJB that your application uses. For information on adding EJBs and declaring them in `application.xml` (and other deployment descriptors), refer to "Add JAR Files" on page 5-27.

Remove any `<ejb>` modules declaring EJBs that you are not including in your enterprise application.

Declarations of Security Roles

The `application.xml` file must declare all J2EE security roles that constituent Web applications use.

A WebLogic Portal `application.xml` file must declare all of the following security roles. Do not modify them: the EJBs and other components that support your applications expect these role names to be present in the application:

```
<security-role>
  <description>Registered customers with role CustomerRole
  </description>
  <role-name>CustomerRole</role-name>
</security-role>

<security-role>
  <description>Portal Administrators</description>
  <role-name>AdminRole</role-name>
</security-role>

<security-role>
  <description>Anonymous access</description>
  <role-name>AnonymousRole</role-name>
</security-role>

<security-role>
  <description>Administrative role for ebusiness</description>
  <role-name>AdministrativeRole</role-name>
</security-role>
```

You can declare additional security roles to support your security scheme.

The `weblogic.xml` file for each Web application maps principals in the security realm to these J2EE security roles. For more information, refer to "Security-Role Mappings" on page 4-22.

The `web.xml` file for each Web application declares which role can access a set of secured resources. For more information, refer to “Declarations of Secure JSPs” on page 4-18.

Modify application-config.xml

The `application-config.xml` deployment descriptor declares MBeans and provides a persistent storage for their configuration data. Before you deploy your enterprise application, use a text editor to remove declarations for MBeans that configure services you do not use.

After you deploy your application, do not use a text editor to modify this file. Instead, use the WebLogic Server Administration Console to modify the MBean configurations. For more information, refer to “Use the WebLogic Server Administration Console to Configure MBeans” on page 5-18.

Similar to the JAR files and supporting Web applications, WebLogic Portal uses specific MBeans to support services. You can remove declarations for MBeans that configure services you do not use.

In all of the following XML elements, you can change the value of the name attribute to match your enterprise application, but it is not required.

The file contains declarations and configurations for the following MBeans:

- Event Service
- Behavior Tracking Service
- Campaign Service
- The Scenario Service
- Ad Service
- Mail Service
- Cache Service
- Document Manager and Document Connection Pool
- Entitlements Service
- JDBC Helper Service
- Tax and Payment Service Clients

Event Service

The following elements set parameters for the Event service:

```
<EventService
  Listeners="com.bea.campaign.internal.CampaignEventListener"
  Name="wlcsApp"
>

</EventService>
```

You can include any of the following values in the Listeners attribute (separate multiple values with a comma):

- `com.bea.p13n.events.listeners.DebugEventListener`, which prints all dispatched events to the command window for debug purposes.
- `com.bea.p13n.tracking.listeners.BehaviorTrackingListener`, which activates the Behavior Tracking service.
- `com.bea.campaign.internal.CampaignEventListener`, which activates campaign-event listening. If you include this value, your application must deploy `campaign.jar`. For more information, refer to “JARs for Campaign Services” on page 5-32.
- A custom Java class that implements `com.bea.p13n.events.EventListener`.

For more information about these properties, refer to "Persisting Tracking Behavior Data" in the *Guide to Events and Behavior Tracking*.

Behavior Tracking Service

The following elements set parameters for the Behavior Tracking service:

```
<BehaviorTracking
    Name="wlcsApp"
    MaxBufferSize="100"
    SweepInterval="10"
    SweepMaxTime="120"
    DataSourceJndiName="weblogic.jdbc.jts.commercePool"

    PersistedEventTypes="AddToCartEvent,BuyEvent,CampaignUserActivityEvent,
    ClickContenEvent,ClickProductEvent,ClickCampaignEvent,DisplayContentEvent,
    DisplayProductProductEvent,DisplayCampaignEvent,PurchaseCartEvent,
    RemoveFromCartEvent,RuleEvent,SessionBeginEvent,SessionEndEvent,
    SessionLoginEvent,UserRegistartionEvent"
>

</BehaviorTracking>
```

Campaign Service

The following element and its subelements configure the Campaign Service for all Web applications in the current enterprise application.

Note: If you include this element and subelements, your `CLASSPATH` variable must specify `campaign_system.jar`. For more information, refer to "Add Directories to CLASSPATH" on page 8-5.

```
<CampaignService
      Name="wlcsApp"
      GoalCheckTime="300000"
      EmailBrowseBaseDir="campaigns/emails"
      EmailURIExtensions="jsp,html,htm,txt"
>

      <CampaignEventListener
        Name="wlcsApp"
       CampaignServiceJNDIName="${APPNAME}.
       BEA_campaign.CampaignService"
      >
      </CampaignEventListener>



      <MailAction
        Name="wlcsApp"
        DefaultFromAddress="acme@acme.com"
        EmailPropertySet="CustomerProperties"
        EmailPropertyName="Email"
        OptInPropertySet="Demographics"
        OptInPropertyName="Email_Opt_In"
       >
      </MailAction>

</CampaignService>
```

For more information about configuring campaigns to send email, refer to "Setting Up and Sending Email for Campaigns" in the *Guide to Developing Campaign Infrastructure*.

For information on configuring basic Mail Service features, refer to "Mail Service" on page 5-13.

The Scenario Service

The following element configures MBeans for the Scenario Service.

Note: If you include this element and subelements, your `CLASSPATH` variable must specify `campaign_system.jar`. For more information, refer to “Add Directories to CLASSPATH” on page 8-5.

```
<ScenarioService
      Name="wlcsApp"
      RulesSessionAttrNames="RulesRequestAttrNames"
>
</ScenarioService>
```

If you use Commerce services with your campaigns, include the following element:

```
<AttributeLoader
 RequestLoaders="com.bea.commerce.ebusiness.campaign.ShoppingCartAttribute
/>
```

In addition to including the `<AttributeLoader>` element your application must deploy `commerce_campaign_bridge_util.jar`. For more information, refer to “JARs for Campaign and Commerce Interaction” on page 5-33.

Ad Service

The attributes and subelements in the `AdService` element support ad placeholders and the `<ad:adTarget>` JSP tag. The `AdContentProvider` element register Java classes that render the HTML necessary to display specific types of content (based on MIME type). If you create Java classes to display additional types of content, you must add a `<AdContentProvider>` element as described in “Supporting Additional MIME Types” under “Setting Up Ads for Campaigns” in the *Guide to Developing Campaign Infrastructure*.

The `AdService` element includes an `EventTracker` attribute. You can place any of the following values in the `EventTracker` attribute (separate multiple entries with a comma):

- `com.bea.p13n.ad.AdEventTrackerBase` (the default).
- `com.bea.campaign.AdTracking`. This attribute enables campaigns to use ad placeholders. If you use this attribute, you must deploy `campaign.jar`.
- A custom written class that implements `com.bea.p13n.ad.AdEventTracker`.

Below is an example of the `<AdService>` element:

```
<AdService
    Name="wlcsApp"
    DisplayFlushSize="10"
    Rendering="com.bea.p13n.ad.AdContentProviderBase"
    EventTracker="<AdEventTracker class name>"
    AdClickThruURI="AdClickThru"
    ShowDocURI="ShowDoc"
>

    <AdContentProvider
      Name="text"
      Provider="com.bea.p13n.ad.render.TextContentProvider"
      Properties=""
     >
     </AdContentProvider>

     <AdContentProvider
       Name="image"
       Provider="com.bea.p13n.ad.render.ImageContentProvider"
       Properties="AdClickThruURI=AdClickThru;ShowDocURI=ShowDoc"
      >
      </AdContentProvider>

     <AdContentProvider
       Name="application/x-shockwave-flash"
       Provider="com.bea.p13n.ad.render.ShockwaveContentProvider"
       Properties="ShowDocURI=ShowDoc"
     >
     </AdContentProvider>

</AdService>
```

Mail Service

The `MailService` element configures the basic features of the Mail Service:

```
<MailService
    Name="wlcsApp"
    SMTPHost="boulder"
>
</MailService>
```

Note: The Campaign Service uses the Mail Service to send email. The `<CampaignService>` element includes a subelement for configuring how campaigns use the Mail Service. For more information, refer to “Campaign Service” on page 5-11.

In addition to using campaigns to access the Mail Service, you can access and use the Mail Service from the `com.bea.p13n.mail.MailManager` command. For more information, refer to “Sending Bulk Mail” in the *Guide to Developing Campaign Infrastructure*.

Cache Service

The `CacheManager` element and its subelements declare named configurations for caches. When you create a cache, you can specify one of these configurations. The `CacheFactory` API spawns an MBean which, in turn, configures the cache on all nodes of a cluster.

For more information about configuring caches, refer to “Using Caches” in the *Performance Tuning Guide*.

Document Manager and Document Connection Pool

Use the `DocumentManager` element to configure the `DocumentManager` EJB, which does the following:

- Gets an enumeration of content objects (documents stored in a content management system) that match the search parameters of a query.
- Retrieves the content objects for the object that initiated the query.

In the following example from a reference `web.xml` file, the Web application configures `DocumentManager` with all default values:

```
<DocumentManager
    Name="default"
    DocumentConnectionPoolName="default"
    PropertyCase="none"
    MetadataCaching="true"
    MetadataCacheName="documentMetadataCache"
    UserIdInCacheKey="false"
    ContentCaching="true"
    ContentCacheName="documentContentCache"
    MaxCachedContentSize="32768"
>
</DocumentManager>
```

Use the `DocumentConnectionPool` element to configure a connection pool for the `DocumentManager` EJB. The `DocumentManager` uses this pool to maintain connections to the content management system, and thus reduce overhead for each time it accesses the system.

In the following example from a reference `web.xml` file, the Web application configures `DocumentConnectionPool`:

```
<DocumentConnectionPool
    Name="default"
    DriverName="com.bea.p13n.content.document.jdbc.Driver"
    URL="jdbc:beasys:docmgmt:com.bea.p13n.content.document.
          ref.RefDocumentProvider"
    Properties="jdbc.dataSource=weblogic.jdbc.pool.commercePool;
                schemaXML=C:/bea/Portal4.0/dmsBase/doc-schemas;
                docBase=C:/bea/Portal4.0/dmsBase"
    InitialCapacity="20"
    MaxCapacity="20"
    CapacityIncrement="0"
/>
```

For more information, refer to "Creating and Managing Content" in the *Guide to Building Personalized Applications*.

Entitlements Service

An entitlement segment is a visitor group based on common characteristics that allows a member of the group to view certain aspects of a portal. For example, if you create a portal that provides information about city council meeting in Los Angeles, you might want to define an entitlement group for that portlet that consists of visitors who live in Los Angeles county and are of voting age.

The Entitlements Service MBean specifies the JNDI name for the Rules Service, which it uses to evaluate whether a visitor belongs to a given entitlement group and to determine which resources the visitor can access.

```
<Entitlements
  Name="wlcsApp"
  RulesServiceJNDIName="BEA_personalization.RulesManager"
>
</Entitlements>
```

Commerce Pipeline

The following element declares an MBean for configuring a JDBC Pool. Commerce Pipeline Components use this JDBC pool to get sequence numbers for new orders (in `CommitOrderPC`) and when generating a transactionID for a payment transaction.

```
<CommercePipelineComponentSupport
      JdbcPoolName="weblogic.jdbc.jts.commercePool"
/>
```

You must deploy this MBean if you use any Order-related Pipeline Components.

If you include this element, your `CLASSPATH` variable must specify `commerce_system.jar`. For more information, refer to “Add Directories to CLASSPATH” on page 8-5.

JDBC Helper Service

The JDBC Helper Service is the only means for services to explicitly establish a database connection. The process for getting a connection includes wait and retry functionality.

The service also provides Java classes (delegates) to coordinate the processing of CLOB data. Use the WebLogic Server Administration Console to specify the delegate class for your specific database type. For information on setting up the JDBC Helper for Oracle and SQL Server database types, refer to Part II, “Deploying Your Business Data.” For information on setting up the service for other database types that are supported for the current release, refer to `PORTAL_HOME/db/readme.html`.

```
<JdbcHelper Name="JdbcHelper"
    JdbcHelperDelegate="@DATABASE_JDBC_HELPER@"
    MaxRetries="-1"
    MaxWaitTime="-1"
>
</JdbcHelper>
```

Tax and Payment Service Clients

The Tax Service client enables your WebLogic Portal application to contact a Web service that will calculate taxes for the orders on your site. WebLogic Portal provides a sample Web service application (taxWSApp). You must find a third party Web service to fulfill this function of your site. The following element in `application-config.xml` configures an MBean, which, in turn, you use to configure the Tax Service client.

You do not need to change this element in `application-config.xml`. Instead, use the WebLogic Server Administration Console to configure the Tax Service client. (Refer to “Use the WebLogic Server Administration Console to Configure MBeans” on page 5-18.)

```
<TaxServiceClient
      Name="wlcsApp"
      TaxCalculatorJNDIName="BEA_commerce.TaxCalculator"
      TaxCalculatorWSDL=
"http://localhost:7501/taxws/BEA_commerce.TaxWebService/wsdl.jsp"
      Currency="USD"
      ShipFromProvince="CO"
      ShipFromCity="Boulder"
      ShipFromPostalCode="80302"
      ShipFromLocationCode="00"
      ShipFromCountryCode="USA"
      OrderOriginProvince="CO"
      OrderOriginCity="Boulder"
      OrderOriginPostalCode="80302"
      OrderOriginLocationCode="00"
      OrderOriginCountryCode="USA"
      CompanyID="Company ID goes here"
      BusinessLocationCode="Business Location code"
      AVSReturnListOnZipFailure="false"
    >
</TaxServiceClient>
```

The `<TaxServiceClient>` element specifies the JNDI name of the tax service (the Order and Payment service use this JNDI name to contact the Tax Service client), the location of your tax Web service application, and default values that WebLogic Portal provides as part of the tax calculation process. For more information on the Tax Service client, refer to “Taxation Services” in the *Guide to Managing Purchases and Processing Orders*.

The Payment Service client enables your WebLogic Portal application to contact a Web service that will confirm and complete credit card transactions for the orders on your site. WebLogic Portal provides a sample Web service application (paymentWSApp). You must find a third party Web service to fulfill this function of your site. The following element in `application-config.xml` configures an MBean, which, in turn, you use to configure the Payment Service client.

You do not need to change this element in `application-config.xml`. Instead, use the WebLogic Server Administration Console to configure the Payment Service client. (Refer to “Use the WebLogic Server Administration Console to Configure MBeans” on page 5-18.)

```
<PaymentServiceClient
      Name="bankApp"
     PaymentWebServiceWSDL=
"http://localhost:7501/paymentws/BEA_commerce.CreditCardWebServic
e/wsdl.jsp"
      DeferAuthorization="true"
      PaymentModel="AUTO_MARK_AUTO_SETTLE"
    >
</PaymentServiceClient>
```

The `<PaymentServiceClient>` element specifies the JNDI name of the payment service (the Order and Payment service use this JNDI name to contact the payment service) and an option to collect all credit card requests as a batch (`DeferAuthorization="true"`). For more information on the Payment Service, refer to “Payment Services” in the *Guide to Managing Purchases and Processing Orders*.

Use the WebLogic Server Administration Console to Configure MBeans

After you deploy your application, use only the WebLogic Server Administration Console to configure MBeans.

To configure MBeans, do the following:

1. Start the WebLogic Server Administration Console by entering the following URL in a Web browser:

   `http://`*`hostname`*`:`*`port-number/`*`console`

   For example, you started a server on a host named bonnie and it uses port 7001 as a listen-on port. Enter the following URL:

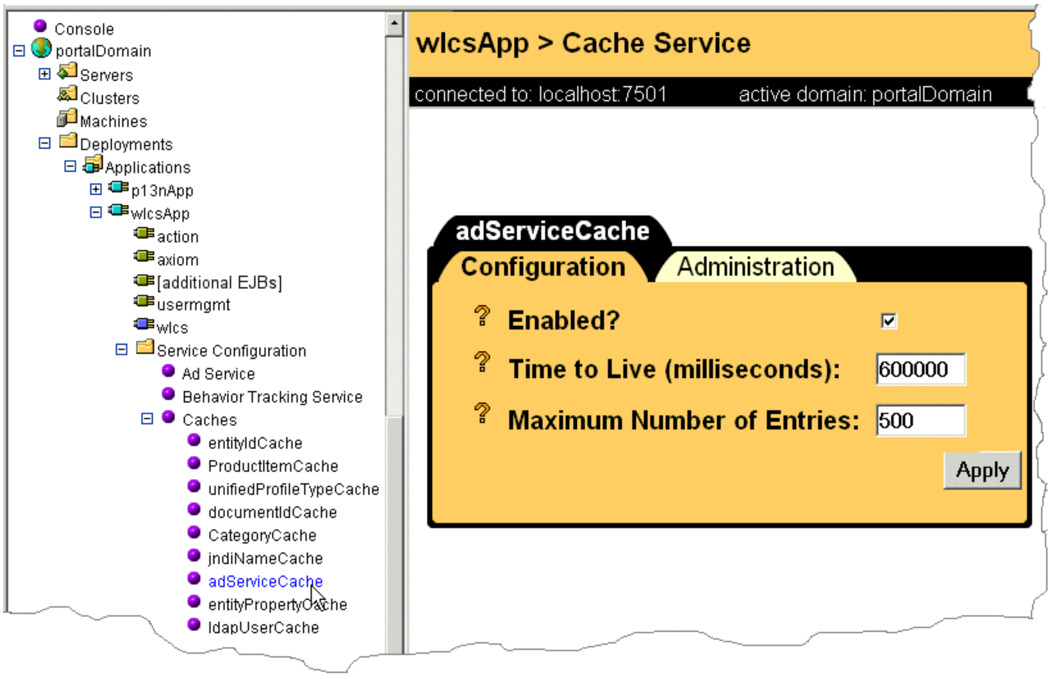
   `http://bonnie:7001/console`

2. The WebLogic Server Administration Console prompts you to log in with a user account that has administrator privileges.

3. After you log in to the WebLogic Server Administration Console, in the left pane, click Deployments → Applications → *MyApplication* → Service Configuration.

4. Click the MBean that you want to configure. (See Figure 5-2 for an example of configuring cache MBeans.)

Figure 5-2 Configuring a Cache MBean



Enabled? ✓

Time to Live (milliseconds): 600000

Maximum Number of Entries: 500

Add Your Web Application

Copy your Web application into the root directory of the enterprise application. In a development environment, we recommend that you deploy the Web application as an exploded directory tree. In this format, you can dynamically deploy additional or modified JSPs and other resources.

In a production environment, deploy the Web application as an archive file (WAR file).

In the `application.xml` deployment descriptor, use the following syntax to declare your Web application:

```
<module>
    <web>
      <web-uri>WAR-file or
      root directory of a non-archived Web application
      </web-uri>
      <context-root>
       name by which you want to access the Web application
       </context-root>
    </web>
</module>
```

Add Supporting Web Applications

Each WebLogic Portal uses supporting Web applications to manage the deployment and administration of application and business data. This section includes the following subsections:

- Adding DataSync
- Adding the WebLogic Portal Administration Tools and ToolSupport
- Adding Support for the E-Business Control Center

Adding DataSync

The DataSync Web application makes data that the E-Business Control Center creates available to one or more enterprise applications. For more information, refer to "How Application Data is Organized and Distributed" on page 7-2.

To add the DataSync Web application, do the following:

1. Copy the `PORTAL_HOME/applications/wlcsApp/datasync` directory and its contents to the root directory of your enterprise application. After the copy, your enterprise application contains the following directory: *`MyApp/`*`datasync`.
2. Look in the *`MyApp/`*`datasync/WEB-INF` directory and remove any directories with names that start with `_tmp_war`. These directories contain compiled versions of the application, which the server generated when you accessed the Web application on the reference site.
3. Modify the URI of the DataSync Web application so that its address reflects the parent enterprise application:
    a. Open *`MyApp/`*`META-INF/application.xml` in a text editor.
    b. Find the XML element that declares the DataSync Web application:
    ```
    <module>
        <web>
          <web-uri>datasync</web-uri>
          <context-root>wlcsAppDataSync</context-root>
        </web>
      </module>
    ```
    c. Modify the value of the `<context-root>` element to provide a unique value. We recommend that you encode the name of the parent enterprise application in the DataSync URI.

    For example, for the Bank enterprise application, you specify `<context-root>BankAppDataSync</context-root>`. Then the following URL accesses the DataSync Web application within the Bank enterprise application: `http://`*`host-name`*`:`*`listen-port`*`/BankAppDataSync`

    BAs use this address in the E-Business Control Center to synchronize their modifications with the `BankApp` application.

Adding DataSync in a Cluster

If you are assembling an enterprise application that you will use in a cluster, you must add two instances of the DataSync Web application to your enterprise application. One instance will be used by Managed Servers for configuring their own Master Data Repositories; the other instance will be used to synchronize data across the cluster.

Note: To reduce the risk associated with synchronizing application data and business policies across a live cluster, you can do the following to synchronize data:

a. Take the cluster offline.

b. Add the DataSync Web applications to your enterprise application.

c. Synchronize the data across the offline cluster.

d. Verify the synchronization.

e. Remove the DataSync Web applications for security purposes.

f. Place the cluster online.

To add two instances of the DataSync Web application, repeat the steps in the previous section, and provide a unique name for each DataSync Web application. For example, use the following names for the DataSync Web applications that you deploy in an enterprise application named Bank:

- `BankAppDataConfig` for the instance that the Managed Servers use to configure their Master Data Repositories.
- `BankAppDataSync` for the instance that synchronizes data across the cluster. Deploy this second Data Sync Web application (the one that synchronizes data) as an exploded directory. Do not archive this Web application until you have configured Proxy Data Repositories as described in “Configuring a Proxy Data Repository” on page 7-14.

For information on configuring and deploying these applications in a cluster, refer to “Deploy and Configure DataSync Web Applications for the Cluster” on page 6-14.

Removing DataSync for Security

To prevent additions or modifications to the data that you have synchronized to your application, you can remove the DataSync Web application after you synchronize. For example:

1. You deploy your enterprise application to your production environment. The application includes the DataSync Web application.
2. You synchronize E-Business Control Center data with the application in your production environment. Note that the application must include DataSync for the synchronization process to succeed.
3. To prevent developers from deploying data to the production environment, you remove the DataSync Web application. If you want to archive your enterprise application, you can do so now.
4. If you want to add or modify data in the application, for example, if you want to modify a discount, you must redeploy the DataSync Web application as described in the previous section. If you archived the enterprise application, you must expand it before you redeploy the DataSync Web application.

For information about undeploying the DataSync Web application in a cluster, refer to “Preventing Synchronization by Undeploying DataSync Web Applications” on page 6-17.

Adding the WebLogic Portal Administration Tools and ToolSupport

Each enterprise application that you deploy must include the WebLogic Portal Administration Tools Web application, which maintains data in the RDBMS repository. For example, you use it to do the following:

- Create and maintain user and group profiles
- Add or remove users from the database security realm
- Maintain catalog items and categories
- Review and manage your customers’ payment and order histories

In addition to the WebLogic Portal Administration Tools, you must also add the ToolSupport Web application.

To add the Web applications, do the following:

1. Copy the `PORTAL_HOME/applications/portal/tools` directory and its contents to the root directory of your enterprise application. After the copy, your enterprise application contains the following directory: `*MyApp*/tools`.
2. Copy the `PORTAL_HOME/applications/portal/toolSupport` directory and its contents to the root directory of your enterprise application.
3. Look in the `*MyApp*/tools/WEB-INF` directory and remove any directories with names that start with `_tmp_war`. These directories contain compiled versions of the applications, which the server generated when you accessed the Web applications on the reference site.
4. Look in the `*MyApp*/toolsupport/WEB-INF` directory and remove any directories with names that start with `_tmp_war`.
5. If you deploy multiple enterprise applications onto the same server instance, you must modify the URI of the WebLogic Portal Administration Tools Web application so that each one uses a unique address.

   To modify the URI, do the following

   a. Open `*MyApp*/META-INF/application.xml` in a text editor.

   b. Find the XML element that declares the WebLogic Portal Administration Tools Web application:

   ```
   <module>
     <web>
       <web-uri>tools</web-uri>
       <context-root>tools</context-root>
     </web>
   </module>
   ```

6. Modify the value of the `<context-root>` element. We recommend that you encode the name of the parent enterprise application in the WebLogic Portal Administration Tools Web application.

   For example, for the Bank enterprise application, you specify `<context-root>tools_BankApp</context-root>`. Then you use the following URL to access the Administration Tools Web application within the Bank enterprise application:

`http://`*`host-name`*`:`*`listen-port`*`/tools_BankApp`

7. In *`MyApp`*`/META-INF/application.xml`, add the following elements to declare `toolSupport`:

```
<module>
    <web>
      <web-uri>toolSupport</web-uri>
      <context-root>wlcsAppTool</context-root>
    </web>
</module>
```

Adding Support for the E-Business Control Center

You can use the E-Business Control Center to browse some types of data that has been deployed to an application. To support this feature, you must deploy a set of helper Web applications:

1. Copy the WAR from the location described in Table 5-1 to the root directory of your enterprise application.

2. Open *`yourApp`*`/META-INF/application.xml` and use the following element syntax to declare the Web application:

```
<module>
  <web>
    <web-uri>
    WAR-file
    </web-uri>
   <context-root>ContextRoot</context-root>
  </web>
</module>
```

Table 5-1 specifies the *`WAR-file`* and for each Web application.

For example, the following elements declare the `catalogws-webservice.war`:

```
<module>
    <web>
      <web-uri>catalogws-webservice.war</web-uri>
      <context-root>wlcsAppCatalogTool</context-root>
    </web>
</module>
```

Table 5-1 Web Applications That Support the E-Business Control Center

Add This Web Application...	By Copying It From...
`propertysetws-webservice.war` Enables the E-Business Control Center to retrieve property sets from the application. If you deploy this Web application, you must also deploy `propertysetws.jar` as described in "JARs for Personalization Services" on page 5-27.	`PORTAL_HOME/lib/p13n/ /web/propertysetws-webservice.war`
`campaignws-webservice.war` Enables the E-Business Control Center to populate ad query and email picklists. If you deploy this Web application, you must also deploy `campaignws.jar` as described in "JARs for Campaign Services" on page 5-32.	`PORTAL_HOME/lib/campaign/ web/campaignws-webservice.war`
`catalogws-webservice.war` Enables the E-Business Control Center browsing the Product Catalog. If you deploy this Web application, you must also deploy `catalogws.jar` as described in "JARs for Commerce Services" on page 5-32.	`PORTAL_HOME/lib/commerce/ web/catalogws-webservice.war`

Add JAR Files

The following sections describe the JARs that WebLogic Portal provides to support your Web application:

- JARs for Personalization Services
- JARs for Campaign Services
- JARs for Commerce Services
- JARs for Campaign and Commerce Interaction
- JARs for Portal Services

Determine which of these JARs you need based on the product you use. Then copy them to the root directory of your enterprise application.

If you created EJBs to extend functionality, copy the JAR that contains them to the root directory of your enterprise application as well.

JARs for Personalization Services

If you use personalization services, do the following:

1. Copy `PORTAL_HOME/lib/p13n/ejb/p13n_util.jar` to the root directory of your enterprise application. This JAR provides miscellaneous utilities for personalization services that you must make available to your application.

2. For each of the JARs that you want to use from Table 5-2, do the following:

   a. Copy the JAR from `PORTAL_HOME/lib/p13n/ejb` to the root directory of your enterprise application.

   b. Open *`yourApp`*`/META-INF/application.xml` and use the following element syntax to declare the EJBs that are in the JAR:
   ```
   <module>
   <ejb>jar-name</ejb>
   </module>
   ```

   c. Open *`yourApp/yourWebApp`*`/WEB-INF/web.xml` and use the following element syntax to specify the EJB interfaces that servlets and JSP tags need to access directly:
   ```
   <ejb-ref>
     <ejb-ref-name>ejb/ServiceName</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <home>home-interface</home>
     <remote>remote-interface</remote>
   </ejb-ref>
   ```
   Table 5-2 indicates *`ServiceName`*, *`home-interface`*, and *`remote-interface`* for each EJB.

   d. Open *`yourApp/yourWebApp`*`/WEB-INF/weblogic.xml` and use the following element syntax to specify the EJB JNDI names that servlets and JSP tags need to access directly:
   ```
   <ejb-reference-description>
   <ejb-ref-name>ejb/ServiceName</ejb-ref-name>
   <jndi-name>${APPNAME}.BEA_personalization.ServiceName
   </jndi-name>
   </ejb-reference-description>
   ```

   Table 5-2 indicates *`ServiceName`* for each EJB.

Table 5-2 Personalization JARs with EJBs That Must Declare References for Web Applications

JAR	Service Name	Interfaces

`document.jar`
Supports document management services (searching, retrieval, and schemas).
	`DocumentManager`	**Home:**
		`com.bea.p13n.content.document.DocumentManagerHome`
		Remote:
		`com.bea.p13n.content.document.DocumentManager`

	`ContentManager`	**Home:**
		`com.bea.p13n.content.ContentManagerHome`
		Remote:
		`com.bea.p13n.content.ContentManager`

`ejbadvisor.jar` Supports Personalization Advisor and advislet framework.	`EjbAdvisor`	**Home:** com.bea.p13n.advisor.EjbAdvisorHome **Remote:** com.bea.p13n.advisor.EjbAdvisor

`events.jar`
Supports the Events and Behavior Tracking service.
	`EventService`	**Home:**
		`com.bea.p13n.events.EventServiceHome`
		Remote:
		`com.bea.p13n.events.EventService`

`ldapprofile.jar`
Provides your application access to user and group profile information that is stored in an LDAP server.
	`LdapPropertyManager`	**Home:**
		`com.bea.p13n.usermgmt.profile.ldap.LdapPropertyManagerHome`
		Remote:
		`com.bea.p13n.usermgmt.profile.ldap.LdapPropertyManager`

Table 5-2 Personalization JARs with EJBs That Must Declare References for Web Applications

JAR	Service Name	Interfaces
`mail.jar` Gives your application access to the outbound Mail Service, which uses the JavaMail API to send email to customers in batches.	`MailService` The sample applications do not declare this service because they do not call the Mail Service directly from any JSPs (instead, the Campaign Service calls the Mail Service). However, if you want to use the Mail Service to generate e-mail from JSPs, you must declare it in `web.xml` and `weblogic.xml`.	**Home:** `com.bea.p13n.mail.MailServiceHome` **Remote:** `com.bea.p13n.mail.MailService`
`pipeline.jar` Support for the Pipeline service.	`PipelineExecutor`	**Home:** `com.bea.p13n.appflow.pipeline.PipelineExecutorHome` **Remote:** `com.bea.p13n.appflow.pipeline.PipelineExecutor`
`placeholder.jar` The placeholder, ads, and ad bucket services.	`AdBucketService`	**Home:** `com.bea.p13n.ad.AdBucketServiceHome` **Remote:** `com.bea.p13n.ad.AdBucketService`
	`AdService`	**Home:** `com.bea.p13n.ad.AdServiceHome` **Remote:** `com.bea.p13n.ad.AdService`
	`PlaceholderService`	**Home:** `com.bea.p13n.placeholder.PlaceholderServiceHome` **Remote:** `com.bea.p13n.placeholder.PlaceholderService`

Table 5-2 Personalization JARs with EJBs That Must Declare References for Web Applications

JAR	Service Name	Interfaces
`property.jar` Provides properties for `com.beasys.commerce.foundation.ConfigurableEntity`. The `ConfigurableEntity` provides the interface to an Entity that can be configured at runtime by associating properties via name-value pairs. The values are permanently associated with the entity by use of the `EntityPropertyManager`.	The EJBs in this JAR do not need to be declared in `web.xml` and `weblogic.xml` to support default WebLogic Portal services.	
`propertysetws.jar` Provides a web service that the `E-Business Control Center` uses to retrieve property set information from an active server.	The EJBs in this JAR do not need to be declared in `web.xml` and `weblogic.xml` to support default WebLogic Portal services.	
`rules.jar` Provides the Rules Manager, which is the public interface to the rules engine.	The EJBs in this JAR do not need to be declared in `web.xml` and `weblogic.xml` to support default WebLogic Portal services.	
`usermgmt.jar` Provides user and group management and Unified User Profile (UUP) services.	The EJBs in this JAR do not need to be declared in `web.xml` and `weblogic.xml` to support default WebLogic Portal services.	

JARs for Campaign Services

If you use campaign services, do the following:

1. Copy the following files from `PORTAL_HOME/lib/campaign/ejb` to the root directory of your enterprise application:
   - `campaignws.jar`. This JAR supports a web service that the E-Business Control Center uses to populate ad query and email picklists.`p13n_util.jar`. This JAR provides miscellaneous utilities for personalization services that you must make available to your application.
   - `campaign.jar`. This JAR includes EJBs that provide campaign and scenario services and repositories.
2. Open *`yourApp`*`/META-INF/application.xml` and add the following elements to declare the campaign EJBs:
   ```
   <module>
   <ejb>campaignws.jar</ejb>
   </module>
   <module>
   <ejb>campaign.jar</ejb>
   </module>
   ```

JARs for Commerce Services

If you use commerce services, do the following:

1. Copy the following file from `PORTAL_HOME/lib/commerce/ejb` to the root directory of your enterprise application:

   `commerce_util.jar`. This JAR provides miscellaneous utilities for commerce services.

2. For each of the JARs that you want to use from Table 5-3, do the following:

   a. Copy the JAR from `PORTAL_HOME/lib/commerce/ejb` to the root directory of your enterprise application.

   b. Open *`yourApp`*`/META-INF/application.xml` and use the following element syntax to declare the EJBs that are in the JAR:

   ```
   <module>
   <ejb>jar-name</ejb>
   </module>
   ```

Table 5-3 Commerce JARs with EJBs

JAR	Description
`catalogws.jar`	Provides a web service that the E-Business Control Center uses to browsing the product catalog. This JAR includes EJBs that provide campaign and scenario services and repositories.
`customer.jar`	Supports `com.beasys.commerce.ebusiness.customer.Customer` entity. `Customer` stores the information required to do business with a customer. It inherits most attributes from the `Person` object and adds the ability to authenticate and to bill for product via a credit card.
`ebusiness.jar`	Supports commerce services, including product catalog, order, tax calculation, shipping, payment, and supporting EJB Pipeline components.

JARs for Campaign and Commerce Interaction

If you use campaigns and commerce services together, copy `PORTAL_HOME/lib/commerce_campaign_bridge/ejb/commerce_campaign_bridge_util.jar` to the root directory of your enterprise application.

This JAR provides the campaign service with access to discounts and the shopping cart.

JARs for Portal Services

If you use portal services, do the following:

1. Copy the following files from `PORTAL_HOME/lib/portal/ejb` to the root directory of your enterprise application:
   - `portal_util.jar`. This JAR provides miscellaneous utilities for portal services.
   - `portal.jar`. This JAR includes EJBs that provide portal services.
2. Open *`yourApp`*`/META-INF/application.xml` and add the following element to declare the campaign EJBs:
   ```
   <module>
   <ejb>portal.jar</ejb>
   </module>
   ```

Create an EAR File (Optional)

In a production environment, we recommend that you archive your enterprise application into an Enterprise Archive (EAR) file and deploy the EAR file.

Before you create an EAR file, do the following:

- Undeploy the DataSync Web applications if you want to prevent additions or modifications to the data that you have synchronized to your application. For more information, refer to “Preventing Synchronization by Undeploying DataSync Web Applications” on page 6-17.
- Archive all constituent Web applications as WAR files.

Use the Java `jar` command to archive your enterprise application. For more information, open a command shell and enter the `jar` command with no arguments. With this syntax, the JDK displays help for the command.

Deploy the Application

Use the WebLogic Server Administration Console to complete the following deployment tasks:

- Deploy an Application in the Exploded Directory Format
- Deploy an Application in the EAR File Format
- Deploy Multiple Instances of an Application

Installing an application via the WebLogic Server Administration Console also creates entries for that application in the domain's configuration file (`/config/`*`domain_name`*`/config.xml`). The domain's Administration Server also generates JMX Management Beans (MBeans) that enable configuration and monitoring of the application and application components.

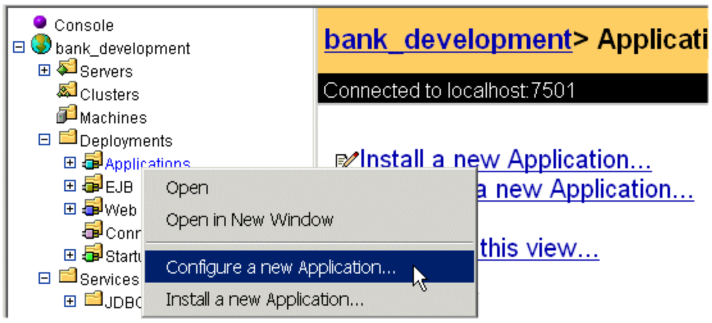
Deploy an Application in the Exploded Directory Format

In a development environment, we recommend that you deploy your enterprise application and Web application in the exploded directory format. To do so, complete these steps:

1. Copy your application onto the computer that hosts the application server.
2. Launch the WebLogic Server Administration Console for the server.
3. In the left pane, open the Deployments folder.

4. Right click the Applications folder and click Configure a New Application. (See Figure 5-3.)

Figure 5-3 Configure a New Application



5. On the Configure a New Application page, do the following:

   a. In the Name box, enter the name of the application. WebLogic Server uses this name only within the WebLogic Server Administration Console. The name does not affect the URIs of resources within the enterprise application.

   b. In the Path box, enter the relative path of the application's root directory.

   c. Select the Deploy check box.

   d. Click Create.

6. In the left pane, open the Servers folder and click the server on which you want to deploy the application.

7. On the Server Page, click the Deployments tab. Then click the Web Applications subtab.

8. On the Web Applications subtab, move the Web applications that are in your enterprise application from the Available to the Selected list. Then click Apply.

9. Click the EJB subtab. Move EJBs from the Available to the Selected list. For information on which EJBs to move, refer to "Add JAR Files" on page 5-27. Then click Apply.

10. Click the Startup/Shutdown subtab. Move any classes from the Available to the Selected list. Then click Apply.

Deploy an Application in the EAR File Format

To facilitate management of your resources in a production environment, we recommend that you deploy your enterprise application as an enterprise application archive (EAR). You cannot add or modify JSPs that are within an archived application. You can, however, use the E-Business Control Center to synchronize application data, use the WebLogic Portal Administration Tools to modify business data, and use the WebLogic Server Administration Console to modify MBean configurations for archived applications.

To archive and deploy an enterprise application, complete the following steps:

1. For each Web application that is in your enterprise application, you must do the following:

   a. Compress the Web application into a Web application Archive (WAR). Use the Java `jar` command to create a WAR file. Make sure that WAR file does not include the name of the Web application's root directory. For example, the pathname for the Web application's deployment descriptor must be `WEB-INF/web.xml` as opposed to *`MyWebApp`*`/WEB-INF/web.xml`

   b. In the deployment descriptor for the enterprise application (`META-INF/application.xml`), change the value of the `<web-uri>` element to refer to a WAR file. For example, you use the following element to declare an exploded Web application: `<web-uri>`*`MyWebApp`*`</web-uri>`. When you archive the Web application, you change the value of the element as follows: `<web-uri>`*`MyWebApp`*`.war</web-uri>`

   c. Remove the exploded Web applications from the enterprise application. After you archive them, you do not need to include the exploded directories in the enterprise application.

   For troubleshooting purposes, after your archive your Web applications, you can deploy your enterprise application as an exploded directory that contains only JAR and WAR files. If you can access your WAR files, then you can move on to the following step.

2. To archive your enterprise application, use the Java `jar` command to create an EAR file. Make sure that EAR file does not include the name of the application's root directory. For example, the pathname for the application's deployment descriptor must be `META-INF/application.xml` as opposed to *`MyApp`*`/META-INF/application.xml`

3. Copy your application onto the computer that hosts the application server.
4. Launch the WebLogic Server Administration Console for the server.
5. In the left pane, open the Deployments folder.
6. Right click the Applications folder and click Configure a New Application.
7. On the Configure a New Application page, do the following:
   a. In the Name box, enter the name of the application. WebLogic Server uses this name only within the WebLogic Server Administration Console. The name does not affect the URIs of resources within the enterprise application.
   b. In the Path box, enter the pathname of the EAR file.
   c. Select the Deploy check box.
   d. Click Create.
8. In the left pane, open the Servers folder and click the server on which you want to deploy the application.
9. On the Server Page, click the Deployments tab. Then click the Web Applications subtab.
10. On the Web Applications subtab, move the Web applications that are in your enterprise application from the Available to the Selected list. Then click Apply.
11. Click the EJB subtab. Move EJBs from the Available to the Selected list. For information on which EJBs to move, refer to “Add JAR Files” on page 5-27. Then click Apply.
12. Click the Startup/Shutdown subtab. Move classes from the Available to the Selected list. Then click Apply.

Deploy Multiple Instances of an Application

In a development environment, we recommend that you deploy one instance of your enterprise application for each installation of the E-Business Control Center. Providing multiple application instances prevents Business Engineers who are using the E-Business Control Center from overwriting each other’s work.

To deploy multiple copies of an application on a single server, do the following:

1. Copy the enterprise application. You place the copy in a different directory or in the same directory as the original but under a different name. If it is archived in an EAR file, expand the archive into an exploded directory.
2. In a text editor, open the `META-INF/application.xml` file.
3. In `application.xml`, provide a unique context root for each Web application by doing the following:
   a. Find the `<web>` subelement that declares the Web application. For example, the following set of elements declare the dataSync Web application:

      ```
      <module>
          <web>
            <web-uri>datasync</web-uri>
            <context-root>datasync</context-root>
          </web>
        </module>
      ```

   b. Change the value for the `<context-root>` element. For example, `<context-root>patDataSync</context-root>`.

   c. Save `application.xml`.
4. Deploy the enterprise application as described in "Deploy the Application" on page 5-35.
5. To access Web applications within the enterprise application, use the value of the `<context-root>` that you specified in step 3. For example, to access a DataSync Web that you deployed with a context root of `datasync-pat`, enter the following URL: `http://`*`hostname:listen-port`*`/patDataSync`

Note: The WebLogic Server Administration Console lists each instance of the Web applications and EJBs with the same name. The server successfully deploys the modules under different JNDI names. If you want to determine the enterprise application within which a particular module resides, click the module in the left pane of the WebLogic Server Administration Console. The right pane displays the pathname to the module.

Configure the JDBC Helper Service

The JDBC Helper Service is the only means for services to explicitly establish a database connection. The JDBC Helper Service enables services to explicitly establish a database connection and to coordinate the processing of CLOB data. To configure the JDBC Helper service for a Cloudscape RDBMS, do the following:

1. Start the WebLogic Server Administration Console for your domain.
2. In the left pane of the WebLogic Server Administration Console, click Deployments → Applications → *yourApplication* → Service Configuration → JDBC Helper Service.
3. In the right pane, on the Configuration tab, enter the following values:

In this box...	Enter this value...
Maximum Number of Retries	The maximum number of times the service attempts to connect to the database. A value of `-1` instructs the service to retry an unlimited number of times.
Maximum Wait Time	The number of milliseconds to wait for a database connection. When the time limit expires, the service attempts another connection, up to the maximum number of retries. A value of `-1` instructs the service to wait infinitely.
Delegate Class Name	Leave empty

4. Click Apply.

Note: For information on configuring a JDBC Helper Service for other database types, refer to Part II, “Deploying Your Business Data.” For Sybase, DB2, and SQL Server 7, the details for setting the JDBC Helper Service are documented in step 2.4 of the section “Update Settings for the JDBC Helper Service” in the <PORTAL_HOME>/db/readme.html file. No Delegate Class Name is required for Cloudscape databases.

6 Deploying Clusters

Clustering provides several advantages that are optimal for Production-type designs including, but not limited to, load-balancing, failover, performance, administration and security. A cluster comprises one Administration Server and one or more Managed Servers. The Administration Server provides most of the configuration for application deployment and network communication, while the Managed Servers handle most of the business logic and client requests.

This topic contains the following sections:

How Clusters Are Organized and Maintained

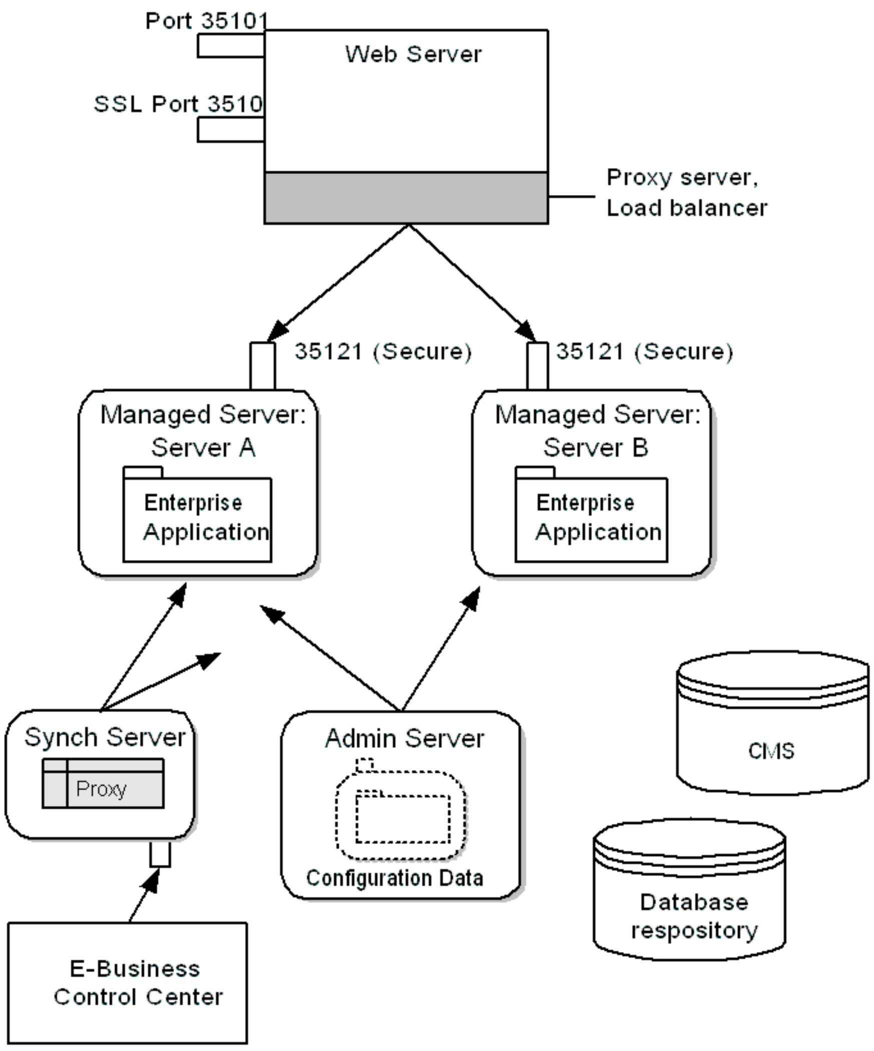
In any WebLogic Server cluster, the host for the Administration Server is the only computer that contains the physical files that describe the cluster, enterprise application, and other supporting services. If you want to modify the cluster or application's configuration, you do so from the Administration Server host. WebLogic Server distributes any configuration changes from the Administration Server to the logical processes and data structures on the Managed Servers.

While WebLogic Server supports a variety of cluster configurations, this topic guides you through deploying a cluster that is in the following recommended configuration (See Figure 6-1):

- The Web server uses a proxy server to direct HTTP requests to the Managed Servers only. You can use a load balancer with your proxy server for a more detailed level of control over HTTP requests.
- The Administration Server is behind a firewall. The proxy server does not forward requests to this server, thus preventing outside access to the server.
- The RDBMS repository, optional content management system, and other back end resources are also behind the firewall, but each node in the cluster maintains its own connection to the resources. The database and content-management systems are outside the purview of WebLogic Server clustering. For information on failover and load balancing these systems, refer to the system vendor.
- The E-Business Control Center deploys data to an additional synchronization server; this server hosts a set of Proxy Data Repositories which forward synchronization request to each Managed Server in the cluster. Consider using a separate computer to host this synchronization server.

  Note: If changes in your production environment are infrequent and highly controlled, or if you want to prevent modifications to the application data in your production environment, you do not need to set up a separate computer to host a synchronization server. For more information, refer to "Preventing Synchronization by Undeploying DataSync Web Applications" on page 6-17.

Figure 6-1 Components of a Cluster



Consider creating at least one mirror site for your cluster. When you need to perform maintenance or deploy additional application data onto a cluster, you can route all customers to a mirror site. Then, when you finish your updates, you can route customers to the original site while you update the mirror.

Before You Start

Before you start to set up a cluster, do the following:

- Install WebLogic Server and WebLogic Portal on each computer in the cluster. For failover purposes, the safest configuration devotes a separate piece of hardware for each Managed Server. At the least, we recommend three separate computers: one for the Administration Server, and two for your Managed Servers. Consider using additional computers as a proxy server and a synchronization server.
- Set up a proxy server. For WebLogic Portal clusters, we recommend that you use third-party Web servers such as Apache as a proxy-server. For these third-party servers, you must set up the WebLogic Server plug-in. For information on the other supported systems, refer to the *WebLogic Server Administration Guide*.
- Set up a database server that hosts your application's RDBMS data repository. All Managed Servers use the same RDBMS server. You cannot use the example Cloudscape RDBMS for a cluster because the trial license allows only one connection at a time. For information on the RDBMS that WebLogic Portal supports, refer to "Supported Platforms" in the *Installation Guide*.
- Set up WebLogic Portal to use your application's RDBMS data repository.

Setting Up a Cluster

To set up a cluster, complete the following tasks on the computer that you want to host the Administration Server:

- Activate the Domain and Start the WebLogic Server Administration Console
- Create Machines
- Create Server Configurations
- Set Up SSL
- Configure a Cluster

Activate the Domain and Start the WebLogic Server Administration Console

Logically, a cluster resides in a single WebLogic Server administrative domain: one domain defines and administers the cluster.

You can create a domain specifically for your cluster or copy an existing domain onto the computer that you want to host the Administration Server. To activate the domain, start an instance of WebLogic Portal on the computer that you want to be the cluster's Administration Server.

Note: Later in the process of setting up a cluster, you will install your enterprise application onto this server. Then WebLogic Server deploys this enterprise application onto the Managed Servers; it will also deploy any subsequent modifications that you make to the enterprise application on the Administration Server.

Then access the WebLogic Server Administration Console for the domain by entering the following URL in a Web browser:

`http://`*`admin-host-name:port-number/`*`console`

For example, `http://localhost:7501/console`

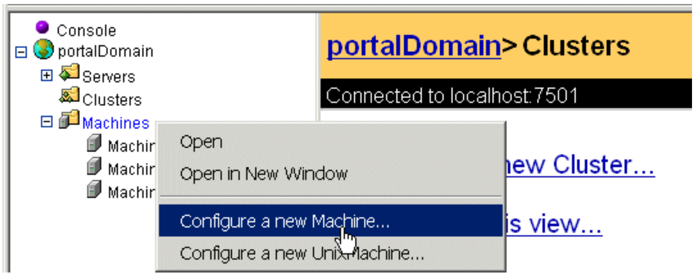
Create Machines

A *machine* is intended to be the logical representation of a single, physical, piece of hardware. WebLogic Server uses machines to establish a preferred order for replicating objects. (Its first preference is to replicate objects on separate machines for failover purposes.)

Create machines for each node by doing the following:

1. In the WebLogic Server Administration Console, in the left pane, right click the Machines folder. Then click Configure a New Machine. (See Figure 6-2.)
2. In the Name box, enter a name for the machine. Use a name that correlates to the name of the server that you intend to deploy onto this piece of hardware. For example, for Server A, enter the name Machine A.
3. Click Create.
4. Repeat until you have created machines for the Administration Server and all Managed Servers in the cluster.

Figure 6-2 Creating a Machine



Create Server Configurations

A cluster contains configurations for one Administration Server and multiple Managed Servers. This section contains the following subsections:

- Creating an Administration Server
- Creating a Synchronization Server

- Creating Managed Servers

Creating an Administration Server

By default, when you start a server instance, it is an Administration Server. You can use the currently active server as the Administration Server for your cluster, or, if you want to create a server configuration specifically for administering your domain, do the following:

1. In the left pane of the WebLogic Server Administration Console, right click Servers.
2. Click Configure a new Server.
3. On the Create a New Server page, in the Name box, enter a name for the server. Use a name that indicates the purpose of the server. For example, `AdminServer`.
4. From the Machine list, select a machine for the server.
5. In the Listen Port box, enter a port number that you will use to connect to the Administration Server. For example, in Figure 6-1 the Administration Server listens on port 35111.
6. In the Listen Address box, enter either the server host's name or IP address. You can use the host name only if you have configured a host file. Entering a host name gives the flexibility of using the hosts file to maintain mappings of host names to IP addresses.
7. Click Create.

Creating a Synchronization Server

To create a server for synchronizing application data across the cluster, do the following:

1. In the left pane of the WebLogic Server Administration Console, right click Servers.
2. Click Configure a new Server.
3. On the Create a New Server page, in the Name box, enter a name for the server. For example, `SynchServer`.

4. From the Machine list, select a machine for the server.
5. In the Listen Port box, enter a port number that the E-Business Control Center will use to connect to the Synchronization Server.
6. In the Listen Address box, enter either the server host's name or IP address. You can use the host name only if you have configured a host file. Entering a host name gives the flexibility of using the hosts file to maintain mappings of host names to IP addresses.
7. Click Create.

Creating Managed Servers

To create Managed Servers, do the following:

1. In the left pane of the WebLogic Server Administration Console, right click Servers.
2. Click Configure a new Server.
3. On the Create a New Server page, in the Name box, enter a name for the server. Use a name that correlates to the function of the server. For example, for a Managed Server, create a server named Server A.
4. From the Machine list, select a machine for the server.
5. In the Listen Port box, enter a port number on which the server listens for requests from the proxy server. **Each server in the cluster should use the same listen address port**. For example, in Figure 6-1 the Managed Servers listen on port 35121.
6. In the Listen Address box, enter either the server host's name or IP address. You can use the host name only if you have configured a host file. Entering a host name gives the flexibility of using the hosts file to maintain mappings of host names to IP addresses.
7. Click Create.
8. Click the Logging tab. From the Stdout severity threshold list, select `Info`. Then click Apply. Informational messages display information about the server joining the cluster, which is useful for diagnosing any problems with the cluster.
9. Repeat these steps for each Managed Server.

Set Up SSL

To set up SSL, do the following for each Managed Server:

1. In the left pane, open the Servers folder.
2. Click a Managed Server.
3. On the Server page, click the SSL tab.
4. On the SSL tab, check the Enabled box. Then provide values for the following fields:
   - SSL Listen Port. Provide the port number that the proxy server uses for secured communication.
   - Server Key File Name, Server Certificate File Name, Server Certificate Chain File. Enter the pathnames to your certificate files.

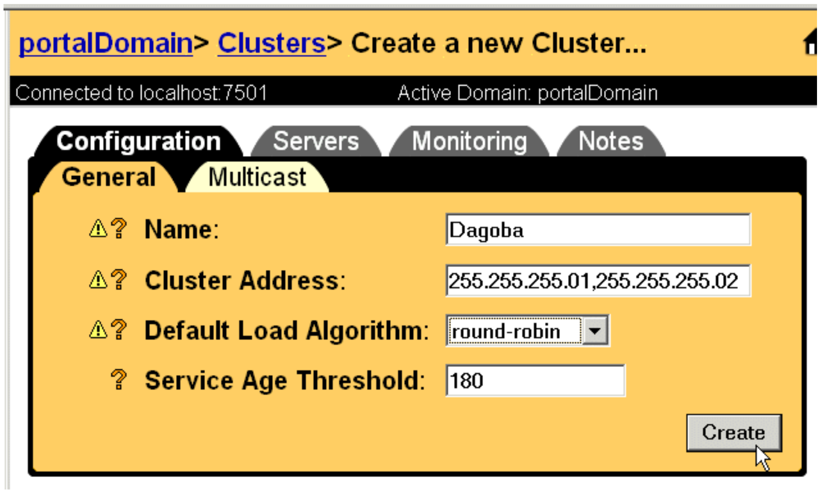
     For example, the reference server specifies the following values:
     Server Key File Name: `config/portalDomain/demokey.pem`
     Server Certificate File Name: `config/portalDomain/democert.pem`
     Certificate Chain File Name: `config/portalDomain/ca.pem`
5. Click Apply.
6. Repeat for the remaining Managed Server configurations.

Configure a Cluster

Configure a cluster by doing the following:

1. In the left pane, right click the Cluster folder.
2. Click Configure a new Cluster.
3. On the Create a New Cluster page, in the Name box, enter a name for the cluster.
4. In the Cluster Address box, enter the IP addresses (or host names) of all the Managed Servers in the cluster. Separate addresses by commas. Do not add the IP address for the Administration Server.
5. Click Create. (See Figure 6-3.)

Figure 6-3 Configure a Cluster



Connected to localhost:7501 Active Domain: portalDomain

Name: Dagoba

Cluster Address: 255.255.255.01,255.255.255.02

Default Load Algorithm: round-robin

Service Age Threshold: 180

6. Set up the Multicast address by doing the following:
    a. On the Cluster page, click the Multicast tab.
    b. In the Multicast Address box, enter an IP address that the nodes in the cluster use to communicate with each other. Multicast addresses must range between `224.0.0.1` and `239.255.255.255`. If you set up multiple clusters on your network, make sure that each cluster uses a unique multicast address.

       We recommend that you conduct a multicast test. For information on using the WebLogic Server `MulticastTest` utility, refer to “Using the WebLogic Java Utilities” in the *WebLogic Server Administration Guide*.
    c. Click Apply.
7. On the Cluster page, click the Servers tab.
8. Move the Managed Servers from the Available list to the Chosen list. Do not place the Administration Server or the Synchronization Server in the Chosen list.
9. Click Apply.

Configure Your Web Application to Support Clusters

Before you deploy your application, configure your Web applications to support clusters by completing the following steps:

1. From the computer that hosts the Administration Server, start a text editor and open your Web application’s `web.xml` file. In the wlcsApp reference application, the file is in the following location: `PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF/web.xml`.

2. The following elements determine which port numbers the `createWebflowURL()` method encodes in the URLs that it generates. Change the value of the `<param-value>` subelements to match the HTTP and HTTPS listen ports of the proxy server:

   ```
   <context-param>
   <param-name>HTTP_PORT</param-name>
   <param-value>35101</param-value>
   </context-param>

   <context-param>
   <param-name>HTTPS_PORT</param-name>
   <param-value>35102</param-value>
   </context-param>
   ```

   This excerpt uses the port numbers from Figure 6-1.

3. Open your Web application's `weblogic.xml` file. In the wlcsApp reference application, the file is in the following location: `PORTAL_HOME/applications/wlcsApp/wlcs/WEB-INF/weblogic.xml`.

4. Add the following elements to enable replication of HTTP Sessions across the cluster nodes:

   ```
   <session-param>
     <param-name>PersistentStoreType</param-name>
     <param-value>replicated</param-value>
   </session-param>
   ```

   Note: The name of the cookie on a proxy server must match the cookie name specified in the `weblogic.xml` of the deployed Web application. For Netscape Enterprise Server, no change is needed. For Apache, you declare the cookie name in the `httpd.conf` file. If WebLogic Server is being used as a Web proxy server, you make the change in the WebLogic Server proxy server's `weblogic.xml` file. For example, the unedited `weblogic.xml` might contain:

   ```
   <session-param>
     <param-name>CookieName</param-name>
     <param-value>
             JSESSIONID
     </param-value>
   </session-param>
   ```

   The wlcsApp, for example, uses `JSESSIONID_WLCS_COMMERCE` in its `weblogic.xml`. On the proxy server's `weblogic.xml`, change the `CookieName` parameter's value to `JSESSIONID_WLCS_COMMERCE`.

For more information on modifying deployment descriptors, refer to the following sections:

- "Listen Ports for Webflow-Generated URLs" on page 4-10 for information on setting up your Web application to listen on the proxy server's port number
- ."Support for Clusters" on page 4-24 for information on supporting the replication of HTTP Sessions across the cluster nodes.

Deploying Applications to a Cluster

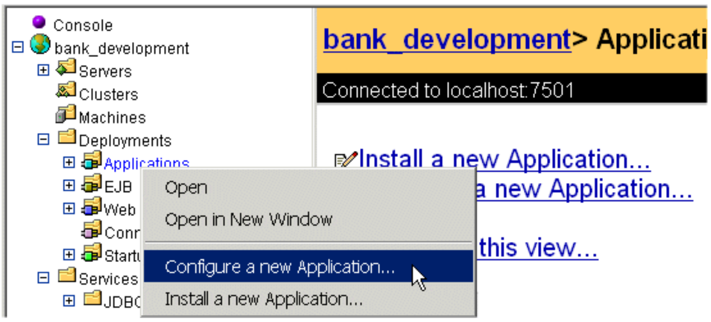
To deploy applications and supporting services to a cluster, complete the following tasks:

- Configure a New Application
- Deploy and Configure DataSync Web Applications for the Cluster
- Deploy Your Web Application and Other Modules to the Cluster

Configure a New Application

1. Copy your application onto the computer that hosts the Administration Server.
2. Launch the WebLogic Server Administration Console for the server.
3. In the left pane, open the Deployments folder.
4. Right click the Applications folder and click Configure a New Application. (See Figure 6-4.)

Figure 6-4 Configure a New Application



5. On the Configure a New Application page, do the following:

   a. In the Name box, enter the name of the application. WebLogic Server uses this name only within the WebLogic Server Administration Console. The name does not affect the URIs of resources within the enterprise application.

   b. In the Path box, enter the pathname of the application's root directory.

   c. Clear the Deploy check box.

   d. Click Create.

Deploy and Configure DataSync Web Applications for the Cluster

The DataSync Web application receives synchronization requests from the E-Business Control Center and channels them to Data Repositories, which are runtime representations of application data.

Clusters with Shared Access to Disk Containing DataSync Database Pool

If your cluster environment provides shared access to the disk containing the DataSync Database Pool, do the following to deploy the DataSync Web application that the cluster uses for configuration purposes:

1. In the left pane, click Deployments → Web Applications and then click the DataSync Web application that the cluster uses for configuration purposes. For example, if you named this Web application `BankAppDataConfig`, click `BankAppDataConfig` in the left pane.
2. On the Web application page, click the Targets tab. Then click the Servers sub tab.
3. Move the Synchronization Server from the Available column to the Chosen column. Then click Apply.
4. Click the Clusters sub tab.
5. Move your cluster from the Available column to the Chosen column. Then click Apply.
6. Deploy the DataSync Connection Pool to these servers (the synchronization server and the managed servers).
7. Sync to the URL of the Synchronization Server.
8. Refresh the data for each of the Managed Servers:
   a. From a Web browser, enter the following URL:

      http://*hostname*:*port-number*/*datasync*/index.html

      Where *datasync* is the name of your DataSync Web application. For more information, refer to "Adding DataSync in a Cluster" on page 5-22.

   b. On the index page, click the Data Repository Browser link.

   c. In the Master Data Repository table, click the Refresh and Notify icon.

Clusters without Shared Access to Disk Containing DataSync Database Pool

If your cluster environment does not provide shared access to the disk containing the DataSync Database Pool, we require that you configure **two** DataSync Web applications for each cluster. You will need:

- A DataSync Web application that synchronizes data for the cluster. The Master Data Repository in this application declares one Proxy Data Repository for each Managed Server in the cluster. A *Proxy Data Repository* forwards synchronization notifications from a Master Data Repository to a DataSync Web application that is located on another physical machine in the cluster.

You deploy this Web application to the Synchronization Server only. If you do not use a Synchronization Server, you can deploy this Web application on any server in the cluster.

- A DataSync Web application that Managed Servers use for configuration information. The Master Data Repository in this application **does not declare Proxy Data Repositories**.

  You deploy this Web application to the cluster. When you start a Managed Server, it instantiates its own DataSync Web application based on the configuration of this Web application. Then it synchronizes its data repositories with the Master Data Repository that is deployed on the Synchronization Server (or Administration Server).

This section contains the following subsections:

- Deploying the Configuration DataSync Web Application
- Deploying the Synchronization DataSync Web Application

For more information about Data Repositories, refer to "Synchronized Data" on page 7-7.

Deploying the Configuration DataSync Web Application

Do the following to deploy the DataSync Web application that the cluster uses for configuration purposes:

1. In the left pane, click Deployments → Web Applications and then click the DataSync Web application that the cluster uses for configuration purposes. For example, if you named this Web application `BankAppDataConfig`, click `BankAppDataConfig` in the left pane.
2. On the Web application page, click the Targets tab. Then click the Clusters sub tab.
3. Move your cluster from the Available column to the Chosen column. Then click Apply.

Deploying the Synchronization DataSync Web Application

Do the following to deploy the DataSync Web application that you use to synchronize data:

1. In the left pane, click Deployments → Web Applications and then click the DataSync Web application that you use to synchronize data across the cluster. For example, if you named this Web application `BankAppDataSync`, click `BankAppDataSync` in the left pane.
2. On the Web application page, click the Targets tab. Then click the Servers sub tab.
3. Move the Synchronization Server from the Available column to the Chosen column. Make sure that the Synchronization Server is the only server in the Chosen column. Then click Apply.
4. Configure Proxy Data Repositories for this Web application. For more information, refer to "Configuring a Proxy Data Repository" on page 7-14.

   When you want to synchronize data for the cluster, you specify the URL of this DataSync Web application. It sends the synchronization information to its proxies, which, in turn, notify the repositories on each Managed Server.

Preventing Synchronization by Undeploying DataSync Web Applications

If changes in your production environment are infrequent and highly controlled, we recommend that after you start a Managed Server (which causes the server to synchronize its Data Repositories with the Master on the Synchronization Server or Administration Server) after you force a refresh-and-notify operation for your Managed Servers, do the following:

1. Shut down the Managed Servers and the Administration Server.
2. On the Administration Server, if the application is archived, expand the archive.

   In a text editor, open the `application.xml` file for your application and remove the declaration for the DataSync Web application. For example, in the sample wlcsApp, open `PORTAL_HOME/applications/wlcsApp/META-INF/application.xml` and remove the following element:

   ```
   <module>
     <web>
       <web-uri>datasync</web-uri>
       <context-root>wlcsAppDataSync</context-root>
     </web>
   </module>
   ```

3. Re-archive the enterprise application and restart the cluster's Administration Server. Then restart the Managed Servers.

For information on forcing a refresh-and-notify operation, refer to "Refreshing the Contents of Data Repositories" on page 7-40.

Configuring Read-Write Persistence

When the DataSync Web application is deployed on the WebLogic Server Synchronization Server for a cluster (or on a single WebLogic Server instance in a single-server environment), the Master Data Repository uses a read-write JDBC Persistence Manager to read and write application data via the dataSync Data Source. The DataSync Web application that you deploy to a Managed Server **reads** data from the database when the server starts (or when you use the Data Repository Browser to explicitly refresh the contents). Then it stores the data in memory for use by the Managed Server. It assumes that the DataSync Web application on the Synchronization Server has already written to the database. You can modify this behavior by using the following `java.lang.System` properties:

- `data.sync.jbdc.master.dr = true`, if you want to use read-write JDBC persistence for the Master Data Repository on a server instance.
- `data.sync.jdbc.read.only.master.dr = true`, if you want to use read-only JDBC persistence for a server instance.

Deploy Your Web Application and Other Modules to the Cluster

To deploy the remaining modules of your enterprise application, do the following:

1. In the WebLogic Server Administration Console, in the left pane, open the Applications folder.
2. Open the folder for the application that you configured in "Configure a New Application" on page 6-13.
3. With the exception of the two DataSync Web applications that you deployed in the previous section, do the following for each remaining module in the enterprise application:

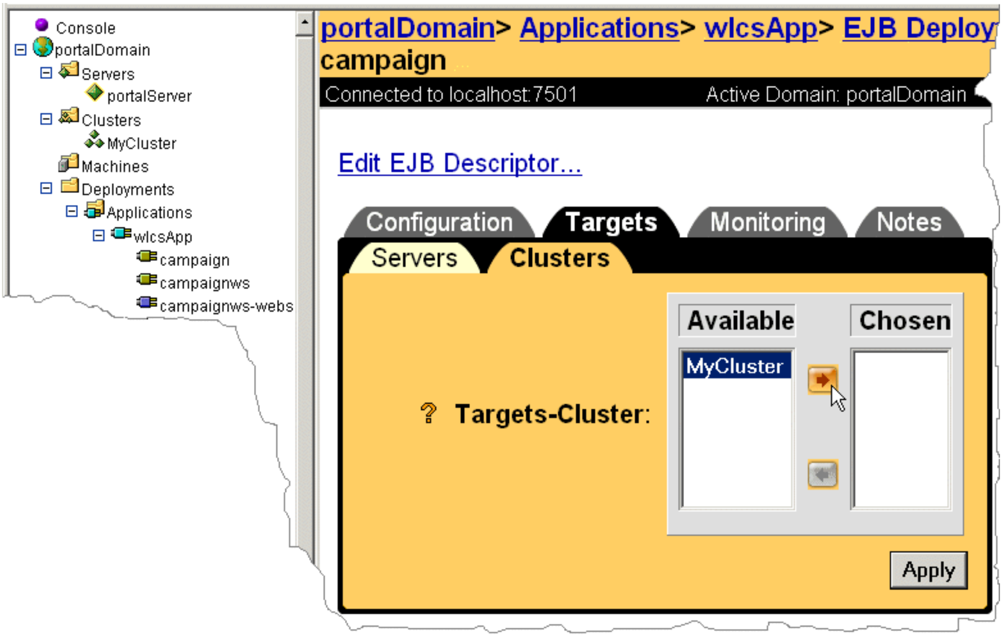
a. In the left pane, select a module.

b. In the right pane, click the Targets tab.

c. Click the Clusters subtab.

d. Move your cluster from the Available to the Chosen list and click Apply. (See Figure 6-5.)

Figure 6-5 Deploy Application Modules to the Cluster



4. In the left pane, under the Deployments folder, open the Startup & Shutdown folder. Do the following for each module in the Startup & Shutdown folder:

a. In the left pane, select a module.

b. In the right pane, click the Targets tab.

c. Click the Clusters subtab.

d. Move your cluster from the Available to the Chosen list and click Apply.

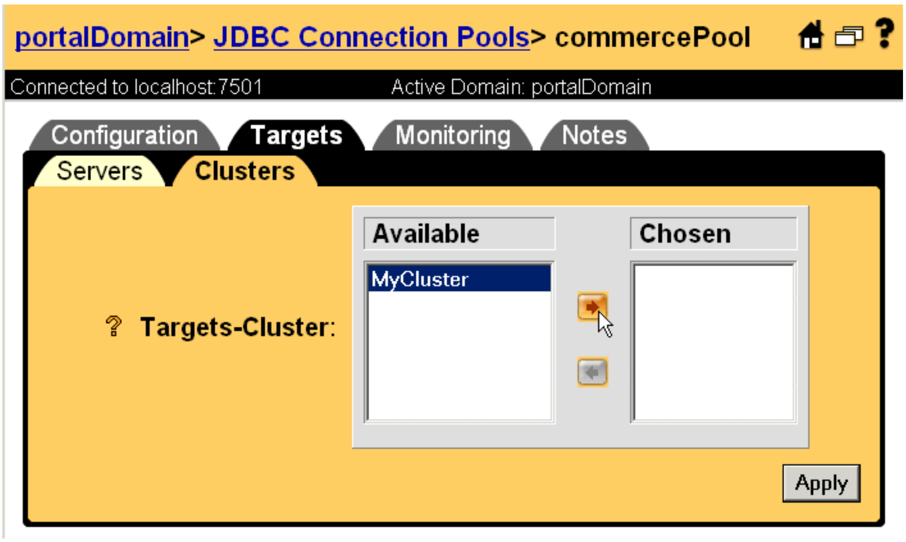
5. In the left pane, click Services → JDBC. For each data pool, data source, and transactional (Tx) data source, do the following:

   a. From the left pane, select a data pool, data source or Tx data source.

   b. In the right pane, click the Targets tab. Then click the Clusters subtab.

   c. Move your cluster from the Available list to the Chosen list and click Apply. (See Figure 6-6.)

Figure 6-6 Deploy Connection Pools to the Cluster



portalDomain> JDBC Connection Pools> commercePool

Connected to localhost:7501 Active Domain: portalDomain

Starting a Cluster

To start a cluster, complete the following tasks:

- Start the WebLogic Server Administration Console
- Start the Synchronization Server
- Start the Managed Servers
- Refresh Data Repositories on Managed Servers

In addition, you must start the proxy server. For information on starting the proxy server, refer to the documentation for the server type that hosts the WebLogic Server plug-in.

Start the WebLogic Server Administration Console

Because the commands for setting environment variables and starting a server are long and prone to typographical errors, we recommend that you use scripts to start servers.

To create a startup script for an Administration Server, copy `PORTAL_HOME/config/portalDomain/startPortal.bat` (`startPortal.sh` on UNIX) onto the computer that hosts the Administration Server. On the target machine you can rename the file, if desired, to a more appropriate name like `AdminSyncServer.bat` (or `.sh`). You should shut down the server before creating a copy of StartPortal.bat and running the new script.

Make the following modifications:

- Change the value for the `SERVER_NAME` variable to match the name of your Administration Server. You specified the name of your Administration Server while "Creating an Administration Server" on page 6-7.
- Change the value for the `DOMAIN_NAME` variable to match the name of the domain on which you created the cluster. It is the domain that "Activate the Domain and Start the WebLogic Server Administration Console" on page 6-5 describes.

Then run the startup script. You must start the Administration Server before the Managed Servers.

Start the Synchronization Server

Because the commands for setting environment variables and starting a server are long and prone to typographical errors, we recommend that you use scripts to start servers.

To create a startup script for a Synchronization Server, copy `PORTAL_HOME/config/portalDomain/startPortal.bat` (`startPortal.sh` on UNIX) onto the computer that hosts the Synchronization Server. On the target machine you can rename the file, if desired, to a more appropriate name like `StartSyncServer.bat` (or `.sh`). Then make the following modifications:

- Change the value for the `SERVER_NAME` variable to match the name of your Synchronization Server. You specified the name of your Synchronization Server while “Creating a Synchronization Server” on page 6-7.
- Change the value for the `DOMAIN_NAME` variable to match the name of the domain on which you created the cluster. It is the domain that “Activate the Domain and Start the WebLogic Server Administration Console” on page 6-5 describes.

Then run the startup script. You must start the Synchronization Server before the Managed Servers.

Start the Managed Servers

To create a startup script for a Managed Server, copy `PORTAL_HOME/config/portalDomain/startPortal.bat` (`startPortal.sh` on UNIX) onto the computer that hosts the Managed Server. On the target machine you can rename the file, if desired, to a more appropriate name like `StartManagedServer.bat` (or `.sh`). Then make the following modifications on each computer:

- Change the value for the `SERVER_NAME` variable to match the name of the Managed Server. You specified the name of a Managed Server while “Creating Managed Servers” on page 6-8.

- Change the value for the `DOMAIN_NAME` variable to match the name of the domain on which you created the cluster. It is the domain that “Activate the Domain and Start the WebLogic Server Administration Console” on page 6-5 describes.
- In the `java` command that starts the JVM, add the following argument:
  - `-Dweblogic.management.server=http://`*`host:port`*

    Where *`host`* is the name or IP address of the machine where the Administration Server is running and *`port`* is the Administration Server's listen port. If you are using Secure Socket Layer (SSL) for communication with the Administration Server, the Administration Server must be specified as:

    `-Dweblogic.management.server=https://`*`host:port`*
  - `-Dweblogic.management.username=`*`system-user-ID`*
    `-Dweblogic.management.password=`*`system password`*

    These username and password arguments are optional. If you do not add them to the startup file, then WebLogic server assumes that the username is `system`, and it prompts you to enter the password on each Managed Server during the server startup process.

For example, add the following variables to the startup script for each Managed Server:

```
SET DOMAIN_NAME=portalDomain
SET SERVER_NAME=ClusterNode1
SET ADMIN_SERVER=192.168.33.137:7501
SET ADMIN_SERVER_USERNAME=system
SET ADMIN_SERVER_PASSWORD=weblogic
```

And then use these variables in the startup command:

```
%JDK_HOME%/bin/java %JAVA_VM% -Xms128m -Xmx128m -classpath %CLASSPATH%
-Dcloudscape.system.home=%WL_PORTAL_HOME%/db/data
-Dweblogic.Domain=%DOMAIN_NAME% -Dweblogic.Name=%SERVER_NAME%
-Dbea.home=%BEA_HOME% -Dweblogic.management.server=%ADMIN_SERVER%
-Dweblogic.management.username=%ADMIN_SERVER_USERNAME%
-Dweblogic.management.password=%ADMIN_SERVER_PASSWORD%
-Djava.security.policy==%WEBLOGIC_HOME%/lib/weblogic.policy
-Dcommerce.properties=%PORTAL_HOME%/weblogiccommerce.properties
-Dpipeline.properties=%PORTAL_HOME%/pipeline.properties
-Dwebflow.properties=%PORTAL_HOME%/webflow.properties weblogic.Server
```

Then run the startup script.

For more information about starting Managed Servers, refer to “Starting a WebLogic Managed Server” under “Starting and Stopping WebLogic Servers” in the *WebLogic Server Administration Guide*.

Refresh Data Repositories on Managed Servers

If you want to refresh the contents of the Data Repositories on the Managed Servers and cause subordinate repositories to refresh their contents, do the following on the Synchronization Server:

1. From a Web browser, enter the following URL:

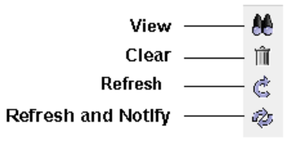
   `http://`*`hostname`*`:`*`port-number`*`/`*`datasync`*`/index.html`

   Where *`datasync`* is the name of your DataSync Web application. For more information, refer to “Adding DataSync in a Cluster” on page 5-22.

2. On the index page, click the Data Repository Browser link.

3. In the Master Data Repository table, click the Refresh and Notify icon.

Figure 6-7 Refresh and Notify



Test Cluster and Application for Failover

To verify that the cluster supports the replication of session data, deploy one of the reference applications onto your cluster. Then access the reference application using the listen port of your proxy server and log in as a sample user. Kill the Managed Server that contains your current session and verify that the session data is available from another Managed Server.

To verify that your application's data is safe in a failover situation, we recommend the following testing methods as a supplement to testing with a multi-node cluster:

- Start a server that is not clustered but that uses JDBC session persistence. Then test for session serialization/deserialization and session management. For example, determine whether the application replaces session objects after you change the values of their member variables. Session objects are serialized each time they are acted upon by a `get`/`set` method and throw exceptions if the serialization fails.
- Set up a single-node cluster (one Administration Server and one Managed Server) and use file persistence instead of in-memory persistence. This configuration serializes sessions without using two nodes and might be useful if your resources are limited.

CHAPTER

7 Synchronizing Application Data

Application data expresses your business model and specifies how your enterprise application behaves to support your business goals. For example, a campaign is a collection of application data that specifies how your enterprise application helps you reach sales or advertising goals.

A Business Engineer (BE) uses the E-Business Control Center to create, modify, and save application data to a local file system. Then a developer uses either the E-Business Control Center or a Java command to synchronize the modified data with an active enterprise application.

A System Administrator can configure the enterprise application to synchronize the data amongst a collection or cluster of active servers and applications.

This topic includes the following sections:

- How Application Data is Organized and Distributed
- Configuring a Proxy Data Repository
- Creating and Modifying Application Data
- Synchronizing Application Data
- Monitoring and Managing Data Repositories

How Application Data is Organized and Distributed

During the development cycle, a Business Engineer (BE) creates application data on a workstation and then synchronizes it with an active enterprise application. Each environment stores the data in different formats:

- On the BE's workstation, the E-Business Control Center stores application data on the local file system as XML files. We recommend that you place these files in your content management system or source control system.
- On the application server, the enterprise application stores the data in the RDBMS database and in data repositories, which provide runtime representations of the data.

This section includes the following subsections:

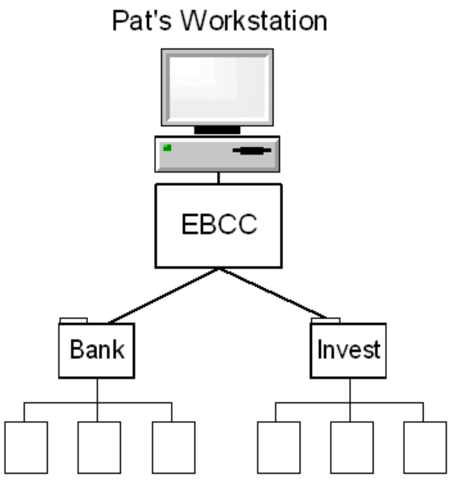
- E-Business Control Center Data
- Synchronized Data

For recommendations on setting up a development environment to support concurrent development, refer to "Milestone 5: Set Up a Development Site" under "Workflow for Developing an E-Business Web Site" in *Strategies for Developing E-Business Web Sites*.

E-Business Control Center Data

The E-Business Control Center creates a separate directory tree of XML files for each application that it modifies. For example, a BE named Pat installs the E-Business Control Center on her workstation and uses it to develop campaigns, customer segments, and portlets for two different enterprise applications, Bank and Invest. (See Figure 7-1.)

Figure 7-1 One Directory Tree for Each Enterprise Application

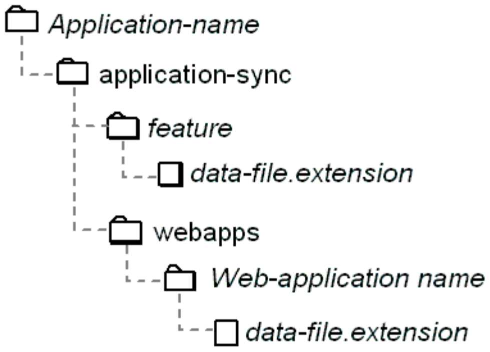


The directory tree of XML files for each application conforms to the following structure (see Figure 7-2):

- The root directory contains two subdirectories: `application-sync`, which contains application data and `library`.
- The `library` directory stores data that the E-Business Control Center uses but does not create or synchronize. For example, `library` contains skin preview images and layout definitions for portals.

- Under `application-sync`, the data for each product feature is in a separate directory. For example, campaigns and customer segments are saved in separate directories (named `campaign` and `segments`).
  - Within each directory, each discrete business policy is saved in a separate file, which uses a specific file extension. For example, each campaign is saved to a separate file with the `.cam` file extension. For a complete list of directories and filename extensions, see Table 7-1.
  - Each Web application within the enterprise application uses a separate folder to save its webflow data. For more information, refer to "Scope of Application Data" on page 7-7.

Figure 7-2 Directory Structure of Client-Side Application Data



For example, for the Bank enterprise application Pat defines two campaigns and two customer segments. (See Figure 7-3.)

Figure 7-3   Example of Application Data on the Client

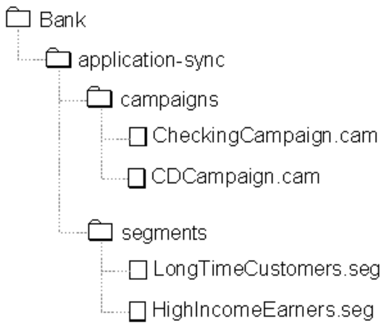


Table 7-1 describes directory and filename extensions for all application data types. All of the directories and files are located under `*root-dir*/application-sync`. Note that `segments`, `contentselectors`, `discounts`, and `entitlements` store all files in a subdirectory.

Table 7-1 Directories and Filename Extensions for Application Data

Feature	Directory Name	Filename Extension
Campaigns	`campaigns`	`cam`
Catalog property sets	`catalog`	`clg`
Content selectors	`contentselectors/GlobalContentSelectors`	`sel`
Discounts	`discounts/DefaultDiscountSet`	`dis`
Entitlements	`entitlements/GlobalEntitlements`	`ent`
Event and Behavior Tracking	`events`	`evt`
Pipelines	`pipelines`	`pln`
Ad placeholders	`placeholders`	`pla`
Portlets	`portlets`	`portlet`
`Request` property sets	`request`	`req`
Scenarios	`scenarios`	`sce`
Customer segments	`segments/GlobalClassifications`	`seg`
`Session` property sets	`session`	`ses`
User profiles	`userprofiles`	`usr`
Webflow	`webapps/`*`web-application`*	`wf`

Scope of Application Data

With the exception of Webflow data, all application data is scoped for an enterprise application. For example, when you synchronize campaign and placeholder data, all Web applications within the enterprise application can use it.

Webflow data, however, is scoped for a single Web application to prevent ambiguities with the Web applications' namespaces. For example, a typical Web application contains a file named `index.jsp`. By scoping Webflow for a single Web application, the filename `index.jsp` is unambiguous: it refers to `index.jsp` in the current Web application.

Synchronized Data

When developers are ready to synchronize data, they can use either the E-Business Control Center or a Java command to synchronize the local files with a deployed application.

This section contains the following subsections:

- The DataSync Web Application and Master Data Repository
- Proxy Data Repositories
- Synchronization Process
- Data Synchronization is Non-Transactional
- Synchronizing Applications That Are Deployed on the Same Server

For instructions on starting a synchronize operation, refer to "Synchronizing Application Data" on page 7-23.

For information on setting up data synchronization in a clustered environment, refer to "Deploy and Configure DataSync Web Applications for the Cluster" on page 6-14.

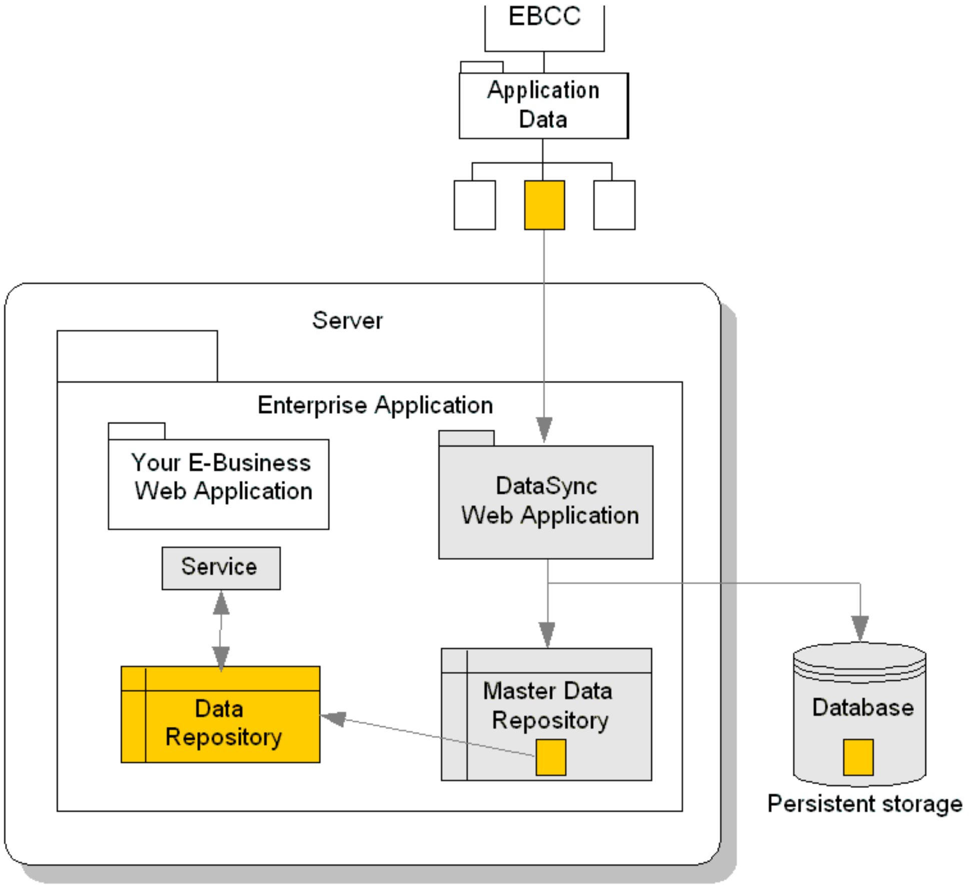
The DataSync Web Application and Master Data Repository

Each enterprise application includes a DataSync Web application, which receives synchronization requests from the E-Business Control Center. After it receives a request, the DataSync Web application channels the application data from the E-Business Control Center to the following locations:

- The Master Data Repository, which is a runtime representation of the application data.
- The RDBMS repository, which is a persistent storage location for the application data.

The services in your enterprise application create their own Data Repositories, which contain subsets of the data that is in the Master Data Repository. (See Figure 7-4.)

Figure 7-4 The Master Data Repository



The services use the data in the Data Repositories to carry out business tasks. For example, the Rules service uses the Data Repositories to evaluate whether customers fit into a particular customer segment.

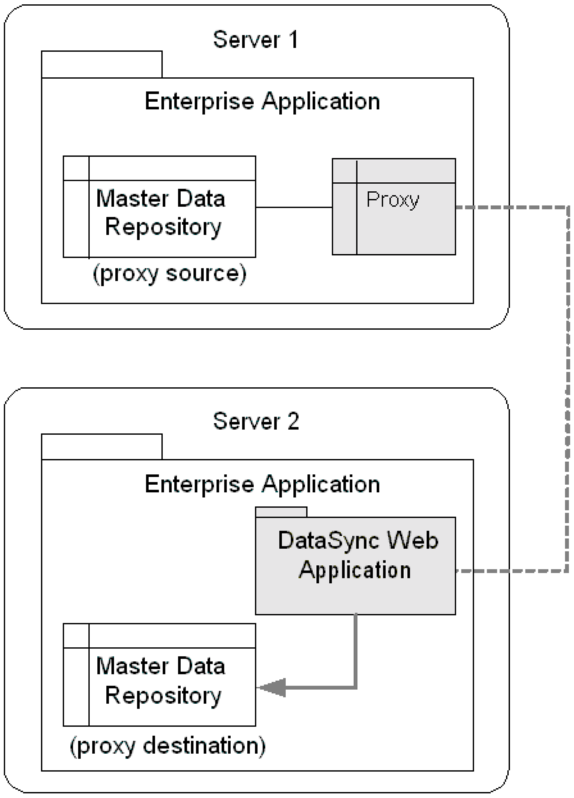
This architecture enables each service to parse a small, specific set of information. For example, the Placeholder Data Repository copies placeholder-related information from the master. When the Placeholder Service requests data about placeholders, it only needs to parse the subset of information in the Placeholder Data Repository.

Each time the Master Data Repository is synchronized, it notifies the Data Repositories. These Data Repositories then synchronize their data subsets with the data in the master.

Proxy Data Repositories

If you want to synchronize the same application data to multiple enterprise applications, you can set up a Proxy Data Repository. A *Proxy Data Repository* represents a notification link to a DataSync Web application that is located in another enterprise application. Usually, you use Proxy Data Repositories to synchronize multiple instances of the same enterprise applications that are deployed on different servers. (See Figure 7-5.)

Figure 7-5 Proxy Data Repository



In a cluster, you can use a Proxy Data Repository to synchronize the Managed Servers. To enhance performance, the Data Repositories on Managed Servers always refer to the Master Data Repository on the proxy source; they never access the persisted data, which is located in the database. Only the Master Data Repository on the proxy source refers to the database when it needs to retrieve data from persistent storage. For more information on setting up Data Repositories for a cluster, refer to “Deploy and Configure DataSync Web Applications for the Cluster” on page 6-14 and “Refresh Data Repositories on Managed Servers” on page 6-24.

In a non-clustered environment, one practical application of a Proxy Server might be to keep an application in a test environment synchronized with an application in the development environment.

Synchronization Process

When a developer synchronizes data, the following process occurs (see Figure 7-6):

1. A developer uses the E-Business Control Center or a Java command to send data files from the local directory tree to a DataSync Web application. The developer determines whether the synchronization operation sends all files from the local directory tree or only the files and directories that have changed.

   To send the XML files to the DataSync Web application, the synchronization operation uses the HTTP protocol.

2. The DataSync Web application updates the Master Data Repository and the RDBMS repository.

3. The Master Data Repository notifies its Data Repositories and any Proxy Data Repositories of the new or modified data.

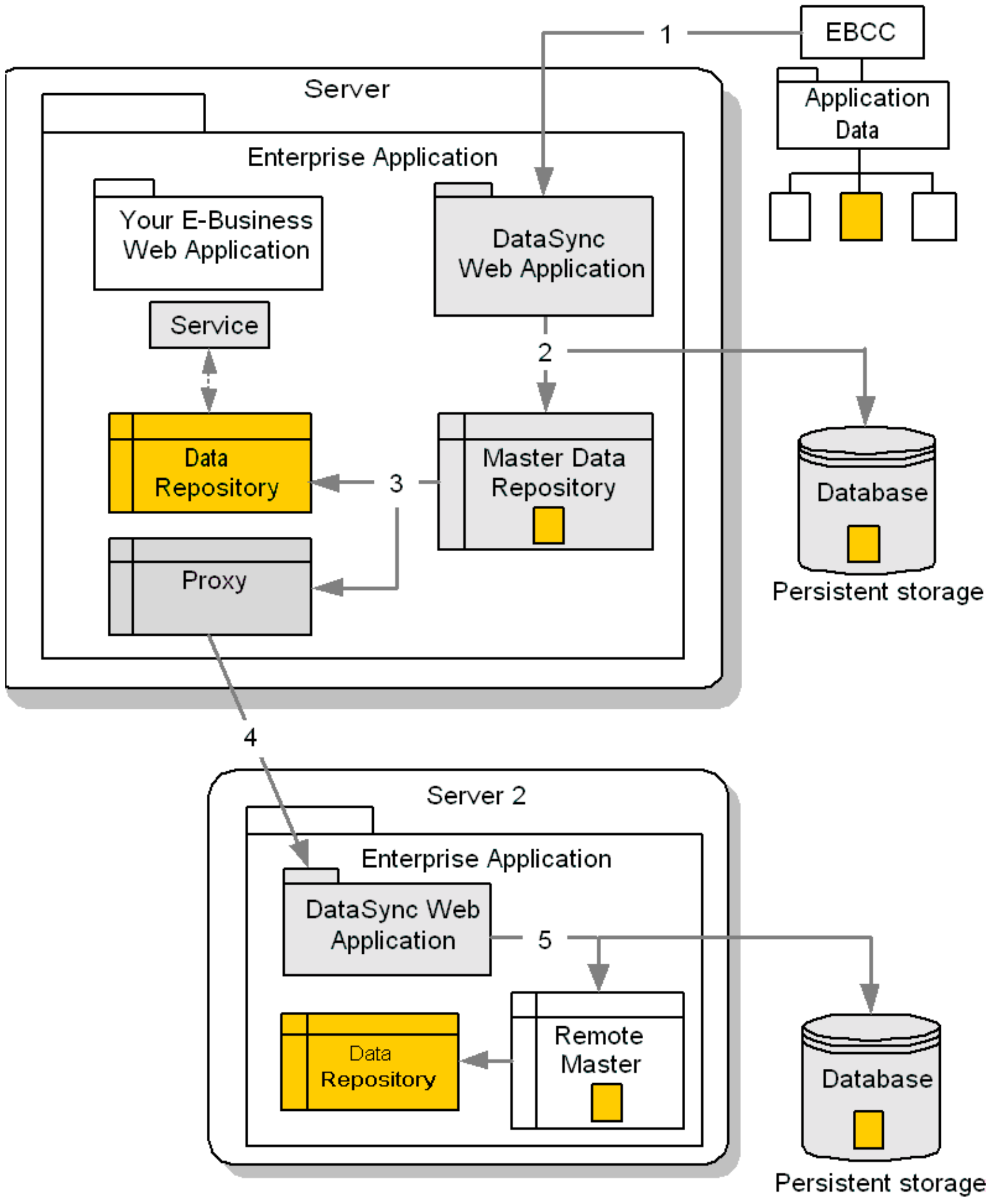
   The notified Data Repositories synchronize themselves with the data that is in the Master Data Repository.

4. The Proxy Data Repositories notify their remote DataSync Web applications.

5. The remote DataSync Web applications synchronize their Master Data Repositories. Then the remote Data Repositories synchronize themselves with the data in their Master Data Repositories.

   The Proxy Data Repositories use XML over HTTP to forward any notification to their remote DataSync Web applications.

Figure 7-6 Synchronizing Data



Data Synchronization is Non-Transactional

The overall process of synchronizing data is non-transactional, though it includes safeguards to prevent data corruption and keeps track of its progress for you to monitor. For more information about monitoring the process of data synchronization, refer to “Monitoring and Managing Data Repositories” on page 7-31.

With the exception of updating the database, if a communications failure (or some other type of problem) prevents the E-Business Control Center or Java command from properly synchronizing one of its files, it does not abandon the synchronization process. Instead, after the process, it reports which files were successfully synchronized and which were not. To prevent data corruption, the synchronization process does not partially synchronize a file. If it cannot synchronize an entire file, it skips the file and moves to the next one in the queue.

Even after the E-Business Control Center transfers data to the DataSync Web application, other communications within and beyond the current application can fail. For example, a Proxy Data Repository might be unable to reach its remote application. If an error occurs during a given transmission, the synchronization continues for other services or machines.

With this process it is possible that one service or machine fails to be synchronized while the others are successfully synchronized. If you use a clustered environment, we recommend that you maintain at least one mirror cluster. When you want to synchronize application data to your production cluster, you can disable one mirror site and synchronize the data onto the disabled site. Before bringing the disabled site back online, you can monitor the data repositories to verify that all subordinate Data Repositories have synchronized. Then repeat the process for the other mirror.

Synchronizing Applications That Are Deployed on the Same Server

Each enterprise application that you deploy must contain its own DataSync Web application. If you deploy multiple enterprise applications onto the same server instance, you must modify the URIs of the DataSync Web applications so that each DataSync uses a unique address. The `application.xml` file includes XML elements that declare the URI for the DataSync Web application. For information on using `application.xml` to declare the DataSync Web application, refer to “Adding DataSync” on page 5-21.

Configuring a Proxy Data Repository

If you want to synchronize the same application data with multiple enterprise applications, you can set up Proxy Data Repositories. This document uses *proxy source* to identify the Master Data Repository that notifies a proxy of synchronization requests; it uses *proxy destination* to identify the remote Master Data Repository to which the proxy forwards the notification. (See Figure 7-5.)

While a typical use for Proxy Data Repositories is to synchronize applications on Managed Servers in a cluster, you can use them to synchronize independent servers that are not participating in a WebLogic Server cluster.

Note: Before you configure Proxy Data Repositories for a cluster, you must deploy two instances of the DataSync Web application. One instance will be used by Managed Servers for configuring their own Master Data Repositories; this instance contains no proxies. The other instance of the DataSync Web application is used to synchronize data across the cluster; this second instance is the Web application in which you configure proxies. For more information, refer to “Deploy and Configure DataSync Web Applications for the Cluster” on page 6-14.

Configuring Proxies

To create a Proxy Data Repository, do the following:

1. Start the WebLogic Server Administration Console by entering the following URL in a Web browser:

   `http://`*`hostname`*`:`*`port-number`*`/console`

2. The WebLogic Server Administration Console prompts you to log in with a user account that has administrator privileges.

3. After you log in, in the left pane, click Deployments → Applications → *MyApplication* → *datasyncWebApp*

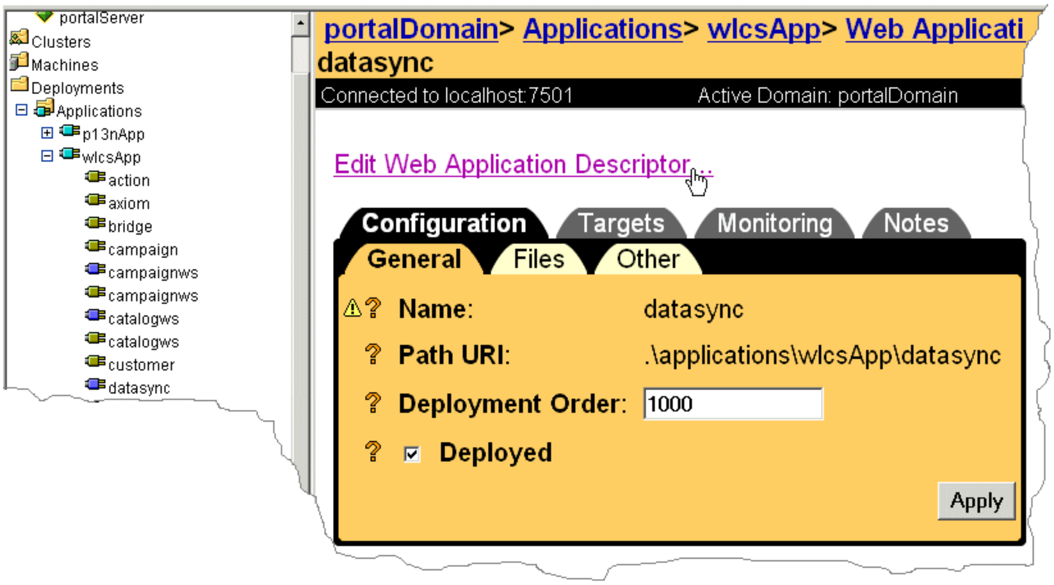
   where *datasyncWebApp* is the DataSync Web application that you deployed for the purpose of synchronizing data. The name of the DataSync Web application depends on how you configured your enterprise application. For information on configuring the DataSync Web applications, refer to “Adding DataSync” on page 5-21.

In the example in “Deploy and Configure DataSync Web Applications for the Cluster” on page 6-14, this server is named `datasyncBankSync`.

4. On the *datasyncWebApp* page click Edit Web Application Descriptor. (See Figure 7-7.)

   The WebLogic Server Administration Console opens the Web Application Descriptor in an additional browser window.

Figure 7-7 Edit Web Application Descriptor



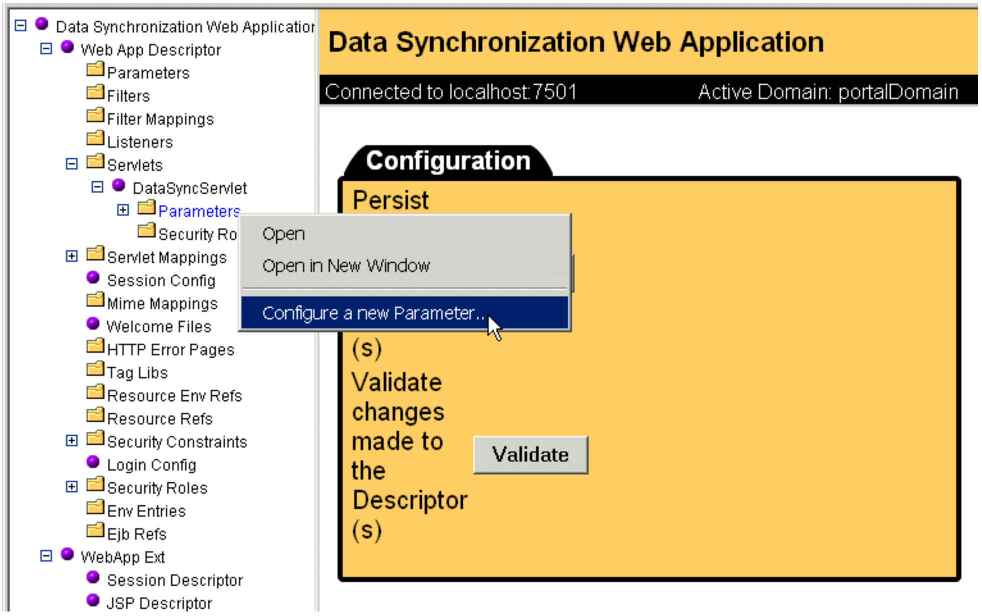
Name: datasync
Path URI: .\applications\wlcsApp\datasync
Deployment Order: 1000
☑ Deployed

5. In the Data Synchronization Web Application window, in the left pane, click Web App Descriptor → Servlets → DataSyncServlet.

6. Complete the following steps for each Managed Server in the cluster:

   a. Right click the Parameters folder and click Configure a New Parameter. (See Figure 7-8.)

Figure 7-8 Configure a New Parameter



b. On the Configure a New Parameter page, enter the following information:

Name	**Value**
Description	The parameter name that you want the WebLogic Server Administration Console to display. We recommend that you adopt a naming convention that includes the following information: ■ Name of the target enterprise application ■ An indication that this parameter configures a Proxy Data Repository ■ The type of parameter. For example: `Bank2ProxyDRnotifiedName`
Param Name	`notifiedName`
Param Value	A name for the Proxy Data Repository.

c. Click Create.

d. Repeat steps a through c for each of the following parameters:

● notifiedURL

Name	**Value**
Description	The parameter name that you want the WebLogic Server Administration Console to display. We recommend that you adopt a naming convention that includes the following information: ■ Name of the target enterprise application ■ An indication that this parameter configures a Proxy Data Repository ■ The type of parameter. For example: `Bank2ProxyDRnotifiedURL`
Param Name	`notifiedURL`

Name	**Value**

Param Value	The URL for the `DataSyncServlet` in the target enterprise application. Use the following syntax:
	`http://`*`remote-host`*`:`*`listen-port/datasyncApp/`*`DataSyncServlet`
	Where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to “Adding DataSync” on page 5-21.
	For example, you want to synchronize the Bank2 enterprise application with the Master Data Repository of the current application. The Bank2 application is deployed on a host named `bread`, which listens on port 7201. The DataSync Web application is named `datasync`. Enter the following URL:
	`http://bread:7201/Bank2DataSync/DataSyncServlet`

- notifiedLogin

Name	**Value**
Description	The parameter name that you want the WebLogic Server Administration Console to display. We recommend that you adopt a naming convention that includes the following information: ■ Name of the target enterprise application ■ An indication that this parameter configures a Proxy Data Repository ■ The type of parameter. For example: `Bank2ProxyDRnotifiedLogin`
Param Name	`notifiedLogin`
Param Value	A user ID that has permission to deploy data.

- notifiedPassword

Name	Value
Description	The parameter name that you want the WebLogic Server Administration Console to display. We recommend that you adopt a naming convention that includes the following information: ■ Name of the target enterprise application ■ An indication that this parameter configures a Proxy Data Repository ■ The type of parameter. For example: `Bank2ProxyDRnotifiedPassword`
Param Name	`notifiedPassword`
Param Value	The password for the user ID.

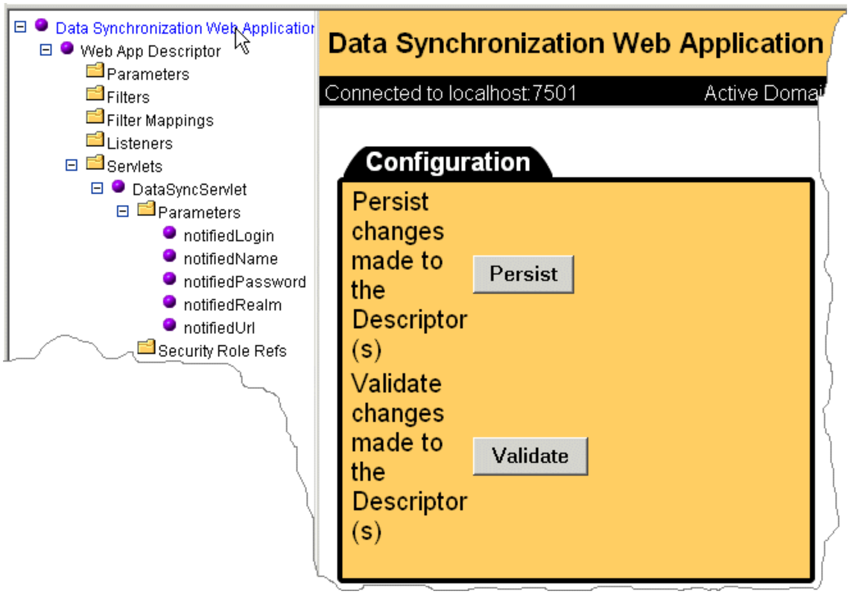
- notifiedRealm

Name	Value
Description	The parameter name that you want the WebLogic Server Administration Console to display. We recommend that you adopt a naming convention that includes the following information: ■ Name of the target enterprise application ■ An indication that this parameter configures a Proxy Data Repository ■ The type of parameter. For example: `Bank2ProxyDRnotifiedRealm`
Param Name	`notifiedRealm`
Param Value	The security realm that defines the user ID for deploying data.

7. In the left pane, click Data Synchronization Web Application (the top node in the left pane). (See Figure 7-9.)

8. In the right pane, click Persist to save your changes.

Figure 7-9 Persist Your Changes



Persist changes made to the Descriptor (s)

Validate changes made to the Descriptor (s)

When you synchronize data with the Master Data Repository, it notifies the Proxy Data Repository that you created.

Creating and Modifying Application Data

Before you use the E-Business Control Center to create and modify application data, you must create a directory tree that organizes the data on your computer.

This section contains the following subsections:

- Create an Application Data Directory Tree
- Opening Data for an Existing Application

For information about the directory tree for application data, refer to “E-Business Control Center Data” on page 7-3.

Create an Application Data Directory Tree

For a new application, you can create the directory tree either through the E-Business Control Center or by using standard commands from your operating system. After you create the directory tree, place it under source control.

If you want to create or modify application data for an existing application, use your source control system to copy the directory tree to your computer. Then open the data in the E-Business Control Center.

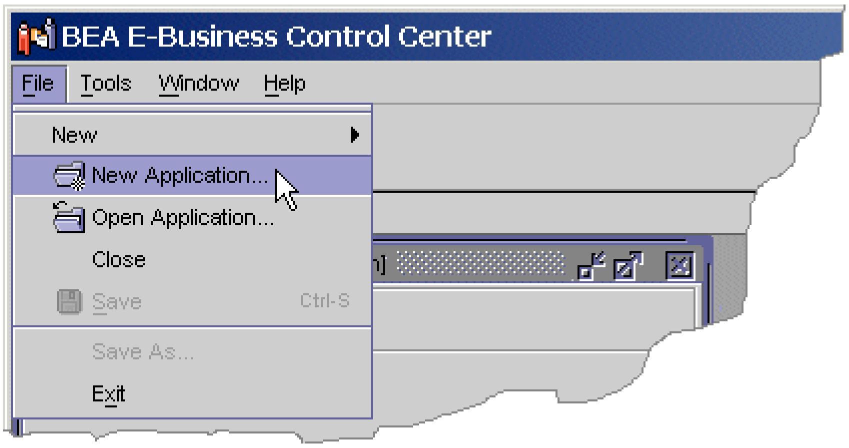
Using the E-Business Control Center to Create a Directory Tree

To use the E-Business Control Center to create a directory tree for saving application data, do the following:

1. Start the E-Business Control Center. For more information, refer to “Introduction to the BEA E-Business Control Center” in *Guide to Using the E-Business Control Center*.

2. In the E-Business Control Center, select File → New Application. (See Figure 7-10.)

Figure 7-10 Select New Application



3. In the Browse Filesystem box, choose a parent directory for your application data. Then in the New Directory box, enter a name for your application-data directory. We recommend that you use the same name as the application.

The E-Business Control Center creates an empty directory tree that conforms to the schema as described in “E-Business Control Center Data” on page 7-3.

Place this directory tree under source control.

Using the Operating System to Create a Directory Tree

You can use standard operating-system commands to create an empty directory tree that conforms to the schema as described in “E-Business Control Center Data” on page 7-3.

Place this directory tree under source control.

Opening Data for an Existing Application

If another developer has already created and checked in an application-data directory tree, do the following to open it on your computer:

1. Use your source control system to copy or update the application directory tree on your computer.
2. If you plan to update existing files, use your source control system to check out the files.
3. Start the E-Business Control Center. For more information, refer to “Introduction to the BEA E-Business Control Center” in *Guide to Using the E-Business Control Center*.
4. In the E-Business Control Center, click File → Open Application.
5. In the Browse Filesystem box, select the root directory of your application-data directory tree. Then click Select.

   Do not select subdirectories within the directory tree.

Synchronizing Application Data

You can use either the E-Business Control Center or a Java command to synchronize application data. This section describes the following tasks:

- Setting Up the E-Business Control Center to Synchronize Data
- Using the E-Business Control Center to Validate Data
- Using the DirectoryDataSync Command to Synchronize Data

Setting Up the E-Business Control Center to Synchronize Data

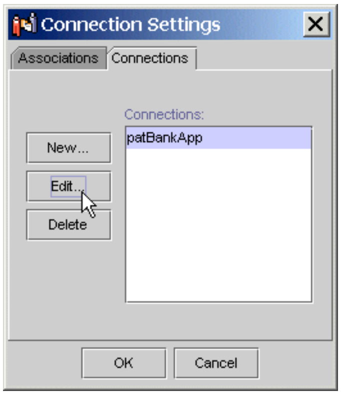
To connect to applications, the E-Business Control Center maintains connection settings. These settings provide such information as the URI of the application with which it exchanges data. Because the URIs differ for each application, create one connection setting for each application instance that you deploy.

This section describes how to modify a connection setting to specify the URI of your DataSync Web application.

1. In the E-Business Control Center, click Tools → Connection Settings.
2. In the Connection Settings box, click. the Connections tab.
3. From the Connections list, select a connection and click Edit. (See Figure 7-11.)

   If the Connections list is empty, click New.

Figure 7-11 Select an Existing Connection



patBankApp

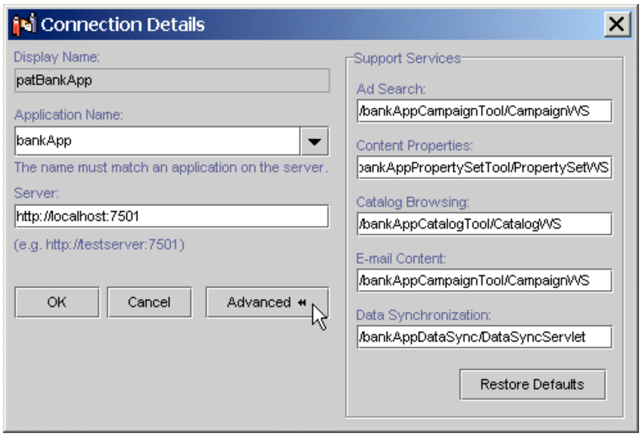
4. In the Connection Details box, click Advanced.

5. In the Data Synchronization box, enter the following (see Figure 7-12):

   `/datasyncApp/DataSyncServlet`

   Where `datasyncApp` is the URI for the DataSync Web application. For more information, refer to “Adding DataSync” on page 5-21.

Figure 7-12 Enter URI for DataSync


Connection Details

Display Name: patBankApp

Application Name: bankApp

The name must match an application on the server.

Server: http://localhost:7501

(e.g. http://testserver:7501)

Support Services

Ad Search: /bankAppCampaignTool/CampaignWS

Content Properties: bankAppPropertySetTool/PropertySetWS

Catalog Browsing: /bankAppCatalogTool/CatalogWS

E-mail Content: /bankAppCampaignTool/CampaignWS

Data Synchronization: /bankAppDataSync/DataSyncServlet

6. Click OK.

For information on the other settings in the Connection Details and Connection Settings boxes, refer to the E-Business Control Center online help or to “Using the BEA E-Business Control Center” under “Introduction to the BEA E-Business Control Center” in the *Guide to Using the E-Business Control Center*.

Using the E-Business Control Center to Validate Data

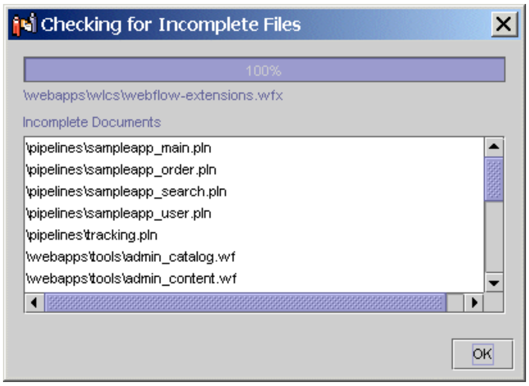
Before you synchronize data to an active application, you can use the E-Business Control Center to scan (validate) the directory tree of data for the current application and report the files that have incomplete data items. For example, you might have saved a discount without defining the conditions under which the discount is available.

To validate an application, do the following:

1. Use your source control system to copy or update the application directory tree on your computer.
2. Start the E-Business Control Center and open the application that you want to validate. For more information, refer to "Introduction to the BEA E-Business Control Center" in *Guide to Using the E-Business Control Center* and "Opening Data for an Existing Application" on page 7-23.
3. When you are ready to start the validation process, click Tools → Validate.

The E-Business Control Center scans the directory tree and reports any incomplete files in the box. You can synchronize data that includes incomplete data items, but you might encounter errors because of the incomplete data. (See Figure 7-13.)

Figure 7-13 Report of Validation Results



\webapps\wlcs\webflow-extensions.wfx

Incomplete Documents

\pipelines\sampleapp_main.pln
\pipelines\sampleapp_order.pln
\pipelines\sampleapp_search.pln
\pipelines\sampleapp_user.pln
\pipelines\tracking.pln
\webapps\tools\admin_catalog.wf
\webapps\tools\admin_content.wf

Using the E-Business Control Center to Synchronize Data

When you are ready to synchronize data with an application on an active server, do the following:

1. Start the E-Business Control Center and open the application that you want to synchronize. For more information, refer to "Introduction to the BEA E-Business Control Center" in *Guide to Using the E-Business Control Center* and "Opening Data for an Existing Application" on page 7-23.
2. Click Tools → Synchronize.

3. In the Synchronization Setup box, click the Synchronization tab and enter the following information:

In this box...	Enter...
Realm	The name of the security realm that contains User IDs for granting permissions to synchronize data. To use the default RDBMS Realm, enter `weblogic`. To use the file realm, enter `fileRealm`.
Scope	Select one of the following: ■ Update only modified files in the application. This option determines which files have been removed, updated or created on the local file system and sends only the changes to the DataSync Web application. Using this option can minimize the amount of data that is transmitted over HTTP between the E-Business Control Center host and the server. ■ Update all files in the application. This option removes all the data from the Data Repositories and then sends all of the data from the local file system to the DataSync Web application. Use this option if there is no data currently synchronized to your enterprise application, if you suspect that the in-memory data is un-synchronized with the persistent data in the database, or to recover from a synchronization error. Before you use this option, make sure that your local file system contains the latest set of source files. (You might need to use your source control system to copy any files that other team members have checked in from their computers.)
Validation	Select any of the following: ■ Check that all files are complete. Scans the directory tree of data for the current application and reports the files that have incomplete data items. This option activates the same validation check as selecting Tools → Validate. You can synchronize data that includes incomplete data items, but you might encounter errors because of the incomplete data. ■ Show reset options for active campaigns. Prompts you to reset such campaign data as the number of times ads have displayed in a particular placeholder or the number of times discounts have been offered.

4. Click the Connections tab and do the following:

   a. From the Connections list, choose the connection that you created or modified in “Setting Up the E-Business Control Center to Synchronize Data” on page 7-24.

   b. In the Username and Password boxes, enter the ID and password of a user who has permission to synchronize data. All users in the `Administrators` group of `wlcsRealm` can synchronize data.

      To add users or other groups to the `Administrators` group, use the WebLogic Portal Administration Tools. For more information, refer to “Using the User Management Tool” under “Creating and Managing Users” in the *Guide to Building Personalized Applications.*

5. Before you click Connect to start the synchronization process, note the following:

   - If files are open in the E-Business Control Center Editor window with unsaved changes when you synchronize, those unsaved changes are not synchronized. Changes must be saved to be synchronized.
   - If you selected any of the validation options, you can stop the process during the validation phase. However, once you start the synchronization process, you cannot cancel or stop it.

6. Click Connect.

Using the DirectoryDataSync Command to Synchronize Data

Before you synchronize data, consider using the E-Business Control Center to validate it. For more information, refer to "Using the E-Business Control Center to Validate Data" on page 7-26.

To synchronize application data to an active server from a command line, enter the following `java` command from an E-Business Control Center host:

```
java com.bea.p13n.management.data.DirectoryDataSync
{ -root pathname-to-directory-where-EBCC-stores-data }
{ -url http://hostname-or-IP:listen-port/datasyncApp/DataSyncServlet }
{ -user user-ID -password user-password }
{ -mode one-way-from-client | refresh-from-client }
```

where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to "Adding DataSync" on page 5-21.

Synchronization Modes

You use the `-mode` argument to determine whether the `DirectoryDataSync` command synchronizes all of the files in the E-Business Control Center directory tree or only the files that have changed. To do so, supply one of the following parameters for `-mode`:

- `refresh-from-client.` This value removes all the data from the Data Repositories and then sends all of the data from the local file system to the DataSync Web application. Use this mode if there is no data currently synchronized to your enterprise application, if you suspect that the in-memory data is un-synchronized with the persistent data in the database, or to recover from a synchronization error. Before you use this command, make sure that your local file system contains the latest set of source files. (You might need to use your source control system to copy any files that other team members have checked in from their computers.)

- `one-way-from-client`. This value determines which files have been removed, updated or created on the local file system and sends only the changes to the DataSync Web application. This synchronization mode can minimize the amount of data that is transmitted over HTTP between the E-Business Control Center host and the server.

For example, the following command synchronizes only the data that you have modified for the Bank application:

```
java com.bea.p13n.management.data.DirectoryDataSync
-root d:\application-data\bank
-url http://255.251.255.01:7501/bankDataSync/DataSyncServlet
-user system -password weblogic -mode one-way-from-client
```

Cancelling the Command

If you cancel the command (by entering Ctrl-C, closing the command shell, or using some other method to force the command to stop), the data that has been synchronized to the application on the server might be in an inconsistent state. The next time you synchronize data, you should use the `-mode refresh-from-client` option.

Monitoring and Managing Data Repositories

The DataSync Web application includes a set of JSPs that report the following information:

- The contents of the Master Data Repository
- The Data Repositories that synchronize themselves with the Master Data Repository
- The contents of each Data Repository
- The Proxy Data Repositories that you associate with the Master Data Repository

In addition, you can use the JSPs to clear and refresh the contents of each repository.

This section contains the following subsections:

- Viewing the Contents of Data Repositories
- Viewing a Notification Chain
- Clearing the Contents of Data Repositories
- Refreshing the Contents of Data Repositories

Viewing the Contents of Data Repositories

To view the contents of Data Repositories, complete any of the following tasks:

- View the Contents of the Master Data Repository
- View the Contents of a Notified Data Repository

View the Contents of the Master Data Repository

The Master Data Repository contains all of the data that was under the root of an application-data directory tree at the time of the last data synchronization operation. To verify that the Master Data Repository contains the same information as the RDBMS repository, refresh the contents. For more information, refer to "Refreshing the Contents of Data Repositories" on page 7-40.

To view the contents of a data repository, a DataSync Web application must be deployed on an active server. After activating the server, do the following:

1. From a Web browser, enter the following URL:

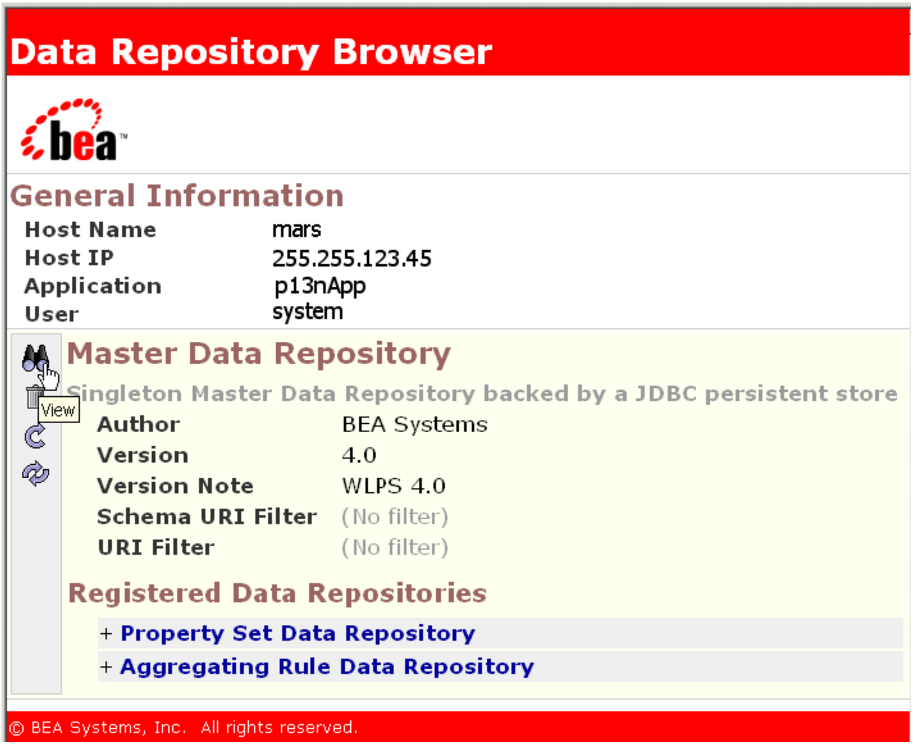
   `http://`*`hostname`*`:`*`port-number/datasyncApp/`*`index.html`

   where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to "Adding DataSync" on page 5-21.

2. On the index page, click the Data Repository Browser link.

3. To view the contents of the Master Data Repository, in the Master Data Repository table, click the View icon. (See Figure 7-14.)

Figure 7-14 View Contents of the Master Data Repository



Data Repository Browser

General Information

Host Name	mars
Host IP	255.255.123.45
Application	p13nApp
User	system

Master Data Repository

Singleton Master Data Repository backed by a JDBC persistent store

Author	BEA Systems
Version	4.0
Version Note	WLPS 4.0
Schema URI Filter	(No filter)
URI Filter	(No filter)

Registered Data Repositories

+ Property Set Data Repository
+ Aggregating Rule Data Repository

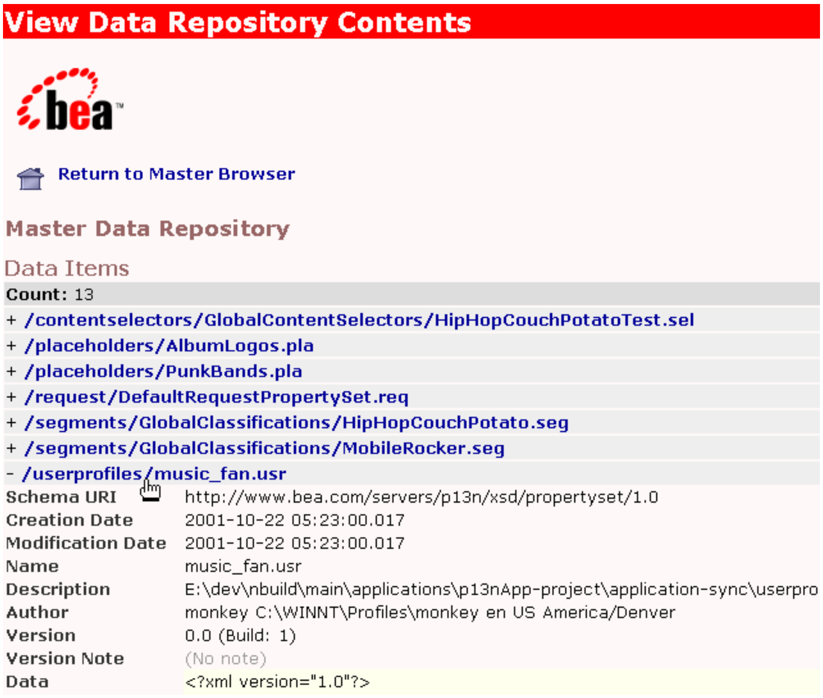
© BEA Systems, Inc. All rights reserved.

The DataSync Web application displays a list of all data items that are in the Master Data Repository.

4. To view the contents of a data item, click it. For example, to see the contents of a property set named `music_fan`, click it. (See Figure 7-15.)

   The Data Repository Browser displays the contents, which is formatted in XML.

Figure 7-15 View the Contents of a Data Item



View Data Repository Contents

Return to Master Browser

Master Data Repository

Data Items

Count: 13

+ /contentselectors/GlobalContentSelectors/HipHopCouchPotatoTest.sel
+ /placeholders/AlbumLogos.pla
+ /placeholders/PunkBands.pla
+ /request/DefaultRequestPropertySet.req
+ /segments/GlobalClassifications/HipHopCouchPotato.seg
+ /segments/GlobalClassifications/MobileRocker.seg
- /userprofiles/music_fan.usr

Schema URI	http://www.bea.com/servers/p13n/xsd/propertyset/1.0
Creation Date	2001-10-22 05:23:00.017
Modification Date	2001-10-22 05:23:00.017
Name	music_fan.usr
Description	E:\dev\nbuild\main\applications\p13nApp-project\application-sync\userpro
Author	monkey C:\WINNT\Profiles\monkey en US America/Denver
Version	0.0 (Build: 1)
Version Note	(No note)
Data	<?xml version="1.0"?>

View the Contents of a Notified Data Repository

Each notified Data Repository contains its own copy of a subset of the information in the Master Data Repository. To verify that a notified Data Repository contains the same information as the Master Data Repository, refresh the contents. For more information, refer to "Refreshing the Contents of Data Repositories" on page 7-40.

To view the contents of the Data Repositories that the Master Data Repository notifies, do the following:

1. From a Web browser, enter the following URL:

   `http://`*`hostname`*`:`*`port-number/datasyncApp/`*`index.html`

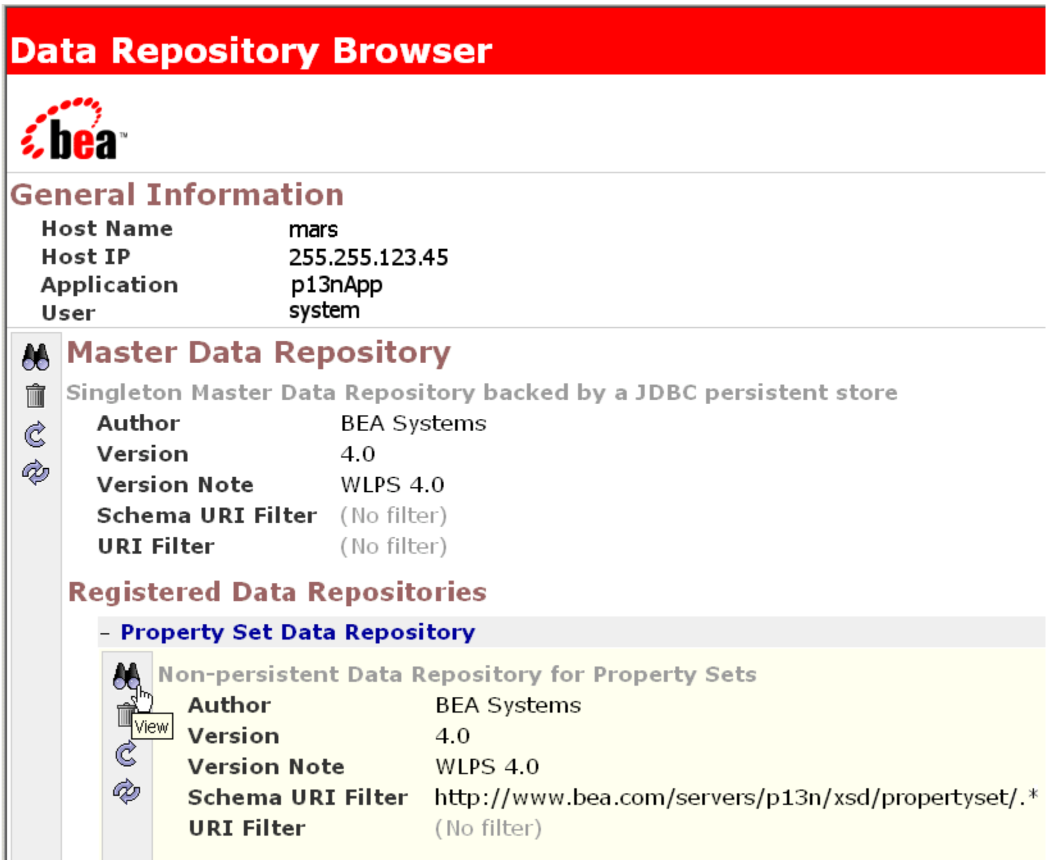
   where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to "Adding DataSync" on page 5-21.

2. On the index page, click the Data Repository Browser link.

3. On the Data Repository page, under Registered Data Repositories, click a repository.

4. To see the contents of the repository, click the view icon under the repository name. (See Figure 7-16.)

Figure 7-16 View the Contents of a Registered Data Repository



Data Repository Browser

General Information

Host Name	mars
Host IP	255.255.123.45
Application	p13nApp
User	system

Master Data Repository

Singleton Master Data Repository backed by a JDBC persistent store

Author	BEA Systems
Version	4.0
Version Note	WLPS 4.0
Schema URI Filter	(No filter)
URI Filter	(No filter)

Registered Data Repositories

– Property Set Data Repository

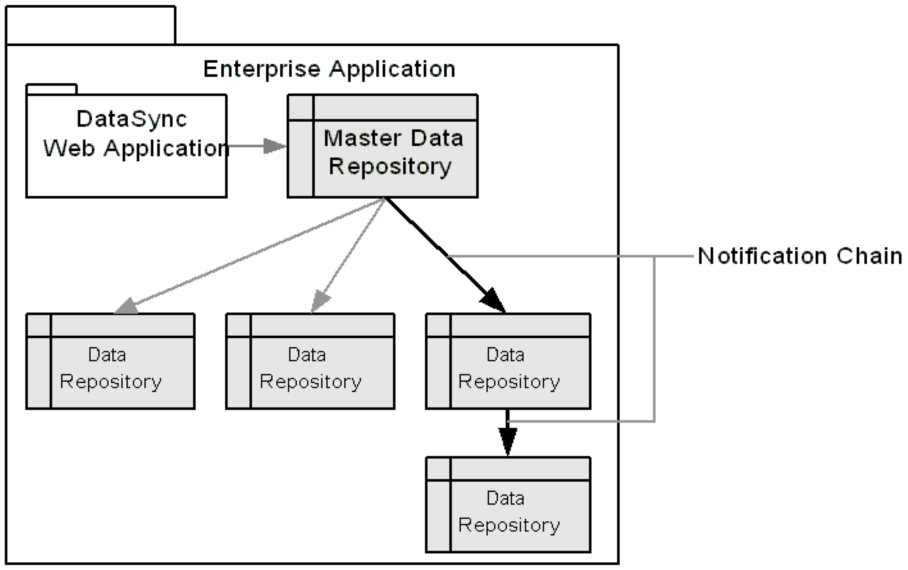
Non-persistent Data Repository for Property Sets

Author	BEA Systems
Version	4.0
Version Note	WLPS 4.0
Schema URI Filter	http://www.bea.com/servers/p13n/xsd/propertyset/.*
URI Filter	(No filter)

Viewing a Notification Chain

In some cases, the Master Data Repository does not directly notify a Data Repository. Instead, some Data Repositories are part of a notification chain: the Master Data Repository notifies one Data Repository, which then notifies another Data Repository. (See Figure 7-17.)

Figure 7-17 Notification Chain



For example, the Master Data Repository notifies the Aggregating Rule Data Repository, which in turn notifies the RuleSet Data Repository. The RuleSet Data Repository then synchronizes its contents with the data in the Aggregating Rule Data Repository.

To view this relationship in the Data Repository Browser, do the following:

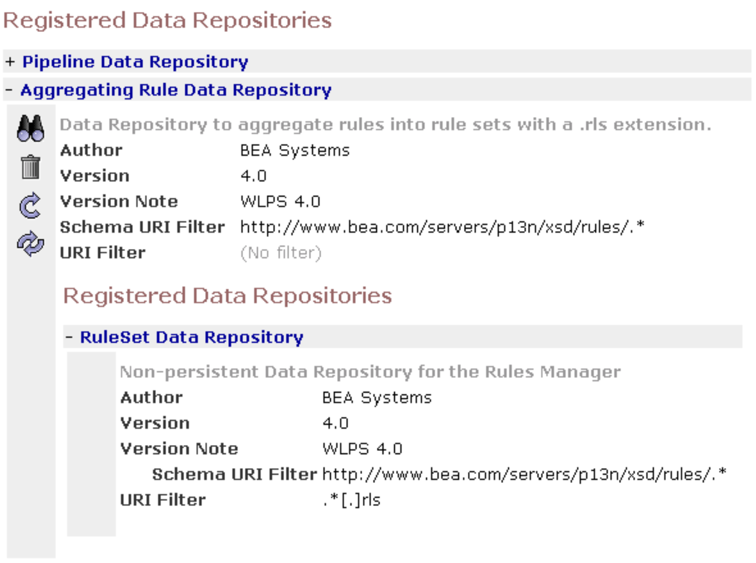
1. From a Web browser, enter the following URL:

   `http://`*`hostname`*`:`*`port-number/datasyncApp/`*`index.html`

   where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to "Adding DataSync" on page 5-21.

2. On the index page, click the Data Repository Browser link.
3. On the Data Repository page, under Registered Data Repositories, click Aggregating Rule Data Repository.
4. Under Aggregating Rule Data Repository, click RuleSet Data Repository. (See Figure 7-18.)

Figure 7-18 Viewing a Notification Chain



Registered Data Repositories

+ Pipeline Data Repository

- Aggregating Rule Data Repository

Data Repository to aggregate rules into rule sets with a .rls extension.

Author	BEA Systems
Version	4.0
Version Note	WLPS 4.0
Schema URI Filter	http://www.bea.com/servers/p13n/xsd/rules/.*
URI Filter	(No filter)

Registered Data Repositories

- RuleSet Data Repository

Non-persistent Data Repository for the Rules Manager

Author	BEA Systems
Version	4.0
Version Note	WLPS 4.0
Schema URI Filter	http://www.bea.com/servers/p13n/xsd/rules/.*
URI Filter	.*[.]rls

Clearing the Contents of Data Repositories

You can use the DataSync Web application to clear the contents of one or more Data Repositories and the corresponding data in the RDBMS repository.

To clear the contents a Data Repository, do the following:

1. From a Web browser, enter the following URL:

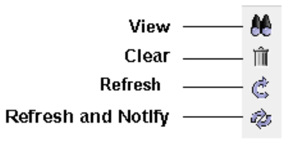
   `http://hostname:port-number/datasyncApp/index.html`

   where *`datasyncApp`* is the URI for the DataSync Web application. For more information, refer to “Adding DataSync” on page 5-21.

2. On the index page, click the Data Repository Browser link.

3. To clear the contents of the Master Data Repository, in the Master Data Repository table, click the Clear icon. (See Figure 7-19.)

Figure 7-19 Clear Contents



After you clear the contents, you must synchronize application data to repopulate the Data Repository.

Refreshing the Contents of Data Repositories

To refresh the contents a Data Repository, do the following:

1. From a Web browser, enter the following URL:

   `http://hostname:port-number/datasyncApp/index.html`

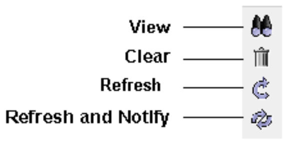
   where `datasyncApp` is the URI for the DataSync Web application. For more information, refer to “Adding DataSync” on page 5-21.

2. On the index page, click the Data Repository Browser link.

3. To refresh the contents of the Master Data Repository, do one of the following:

   - If you do not want notified repositories to refresh their contents, in the Master Data Repository table, click the Refresh icon. (See Figure 7-20.)
   - If you want the notified repositories to synchronize their contents from the Master, click the Refresh and Notify icon that is on the Master Data Repository.

Figure 7-20 Refresh Contents



CHAPTER

8 Starting and Shutting Down a Server

WebLogic Portal provides startup scripts for each of its reference domains. We recommend that you create scripts similar to these to activate and shut down servers in your own domain.

This topic includes the following sections:

- Starting portalDomain on UNIX
- Starting portalDomain on Windows
- Startup Confirmation
- Setting Environment Variables
- Java Command for Starting a Server
- Shutting Down a Server

This topic describes starting and stopping portalDomain because this domain provides the most detailed example of WebLogic Portal features. For information on starting the other reference domains, refer to “Review the Reference Domains” on page 3-4.

Starting portalDomain on UNIX

From a WebLogic Portal host, enter the following command where `PORTAL_HOME` is the directory into which you installed WebLogic Portal:
`PORTAL_HOME/StartPortal.sh`

The `StartPortal.sh` calls `PORTAL_HOME/config/portalDomain/startPortal.sh`, which calls `set-environment.sh` to set environment variables. Then it passes to the JVM the class name and parameters that start WebLogic Server, WebLogic Portal, and activate the portalDomain and its servers.

For information on starting a server without using `StartPortal.sh` (for example, if you want to create a script for your own environment), refer to the following sections:

- Setting Environment Variables
- Java Command for Starting a Server

Starting portalDomain on Windows

From a Windows WebLogic Portal host, do one of the following:

- Click Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Portal 4.0 → Start BEA WebLogic Portal
- From a command prompt, enter the following command:
  `PORTAL_HOME\StartPortal.bat`

The Start WebLogic Portal Server command on the Start menu is a shortcut to `StartPortal.bat`. The `StartPortal.bat` calls `PORTAL_HOME/config/portalDomain/startPortal.bat`, which calls `set-environment.bat` to set environment variables. Then it passes to the JVM the class name and parameters that start WebLogic Server, WebLogic Portal, and activate portalDomain and its servers.

For information on starting a server without using `StartPortal.bat` or the Start WebLogic Portal Server command (for example, if you want to create a script for your own environment), refer to the following sections:

- Setting Environment Variables
- Java Command for Starting a Server

Startup Confirmation

When you issue a startup command, WebLogic Portal displays messages in the shell that contains the server process. The following three messages indicate that the server started successfully:

```
< DATE > <Notice> <WebLogicServer> <ListenThread listening on port NUMBER>

< DATE > <Notice> <WebLogicServer> <SSLListenThread listening on port NUMBER>

< DATE > <Notice> <WebLogicServer> <Started WebLogic Admin Server "server-name"
for domain "domain-name">
```

For information about changing the number and type of messages that WebLogic Portal displays in the shell and saves to log files, refer to “Configure the Message Output” on page 3-11.

Setting Environment Variables

Before starting the server, you must set environment variables and add directories to the system path. Although the `set-environment` file does this for you, you can set the variables in any other way that your operating system supports.

This section describes the following tasks:

- Create New Environment Variables
- Add Directories to CLASSPATH
- Add Directories to the System PATH

Create New Environment Variables

Create the following new environment variables:

Note: The examples in the following list are from a Windows `set-environment.bat` file.

- `PORTAL_HOME=`{ Directory into which you installed WebLogic Portal. }
- `JDK_HOME=`{ Directory into which you installed the JDK that WebLogic Portal requires. }
- `BEA_HOME=`{ Directory that contains BEA license files. }
- `WEBLOGIC_HOME=`{ Directory into which you installed WebLogic Server. }
- `WLCS_ORACLE_HOME=`{ If you are using Oracle, set this variable to the directory in which you installed the Oracle client software. }

Add Directories to CLASSPATH

Add the following directories to the `CLASSPATH` variable:

Note: For ease of maintenance, the `set-environment` file groups these directories into several variables. Then it adds the variables to the `CLASSPATH`.

- If you use the Cloudscape sample database, the location of the Cloudscape JAR files. For example,
  `%WEBLOGIC_HOME%\samples\eval\cloudscape\lib\cloudscape.jar;%WEBLOGIC_HOME%\samples\eval\cloudscape\lib\tools.jar;%WEBLOGIC_HOME%\samples\eval\cloudscape\lib\client.jar`
- The pathname of the `tools.jar` file in the JDK that WebLogic Portal requires. For example, `JDK_TOOLS=%JDK_HOME%\lib\tools.jar`
- The classes that boot WebLogic Server.
  For example,
  `%JDK_TOOLS%;%WEBLOGIC_HOME%\lib\weblogic_sp.jar;%WEBLOGIC_HOME%\lib\weblogic.jar;%WEBLOGIC_HOME%\lib\xmlx.jar`
- The classes required for the WebLogic Portal run-time environment.
  For example,
  `%PORTAL_HOME%\lib\jdom.jar;%PORTAL_HOME%\lib\HTTPClient.jar`
- The system-scoped classes for the WebLogic Portal services that you use (if you are unsure which services you use, add all of the following classes to `CLASSPATH`)
  - `PORTAL_HOME/lib/p13n_system.jar`
  - `PORTAL_HOME/lib/campaign_system.jar`
  - `PORTAL_HOME/lib/commerce_system.jar`
  - `PORTAL_HOME/lib/portal_system.jar`

Add Directories to the System PATH

Add the following directories to the system PATH variable:

- The directory that contains the WebLogic Server binary files.
  For example, `%WEBLOGIC_HOME%\bin`
- If you are using the Oracle jDriver, the directory that contains the driver and the directory that contains the Oracle client binary files.
  For example, `%WEBLOGIC_HOME%\bin\oci817_8;%WLCS_ORACLE_HOME%\bin`

Java Command for Starting a Server

WebLogic Server is a Java class file, and like any Java application, you can start it with the `java` command. The arguments needed to start WebLogic Server with WebLogic Portal classes from the command line can be quite lengthy and typing it out whenever you need to start the server can be tedious.

You must include the following arguments to start an instance of a WebLogic Administration Server that includes WebLogic Server with WebLogic Portal classes:

For Windows:

- `-hotspot`

  or
- `-client`

  Invokes the HotSpot Client VM. For more information, see “Use the HotSpot Virtual Machine” in the *Performance Tuning Guide*.

  Note: For the default Cloudscape database, Windows uses `-classic` as a default.

For Linux or Solaris:

- `-hotspot`

  or

- `-client`

  or

- `-server`

  Invokes the HotSpot Client VM or Server VM.

For HP-UX or AIX:

- `-hotspot`

  or

- `-client`

  Invokes the HotSpot Client VM.

  Note: WebLogic Portal does not specify this setting, so it defaults to the Client VM.

- `-Xms128m -Xmx128m`

  Specifies the initial and maximum values for Java heap memory. The values assigned to these parameters can dramatically affect the performance of WebLogic Server and are provided here only as general defaults. In a production environment you should carefully consider the correct memory heap size to use for your applications and environment.

- `-classpath %CLASSPATH%` (Windows)
  `-classpath $CLASSPATH` (UNIX)

  Sets the `java -classpath` option. The minimum content for this option is described in "Add Directories to CLASSPATH" on page 8-5.

- If you use the Cloudscape database, include the following argument to provide the database location:
  `-Dcloudscape.system.home=%PORTAL_HOME%/db/data`

- `-Dweblogic.Domain=`*`DOMAIN_NAME`*

  Where *`DOMAIN_NAME`* is the name of the domain that contains your Web applications. The domain name must match the name of a directory whose parent directory is named `config`. In addition, the domain directory must contain a `config.xml` file that conforms to the WebLogic domain-configuration DTD.

For example, if your domain is located in `/usr/config/myPortalDomain`, then use the following argument:
`-Dweblogic.Domain=myPortalDomain`

- `-Dweblogic.Name=`*`SERVER_NAME`*

  Where *`SERVER_NAME`* is a logical name of the server instance. When a WebLogic Managed Server requests its configuration information from the Administration Server, it identifies itself to the Administration Server by server name.

- If you are starting a Managed Server, include the following argument:
  `-Dweblogic.management.server=`*`host:port`* or
  `-Dweblogic.management.server=http://`*`host:port`*

  Where *`host`* is the name or IP address of the machine where the Administration Server is running and *`port`* is the Administration Server's listen port. By default the listen port for WebLogic Portal Administration Servers is 7501. For more information about starting Managed Servers, refer to “Starting a WebLogic Managed Server” under “Starting and Stopping WebLogic Servers” in the *WebLogic Server Administration Guide*.

- `-Dbea.home=BEA_HOME`

  Where `BEA_HOME` is the directory that contains BEA license files.

- `-Djava.security.policy==`*`WEBLOGIC_HOME`*`/lib/weblogic.policy`

  Where *`WEBLOGIC_HOME`* is the directory into which you installed WebLogic Server. This argument specifies the Java security policy for the server.

- `-Dcommerce.properties=PORTAL_HOME/weblogiccommerce.properties`

- `weblogic.Server`

  Where `PORTAL_HOME` is the directory into which you installed WebLogic Portal. These arguments load WebLogic Portal classes.

For example, the StartPortal script for an Administration Server on Windows issues the following command:

```
%JDK_HOME%\bin\java -hotspot -Xms128m -Xmx128m -classpath %CLASSPATH%
-Dcloudscape.system.home=PORTAL_HOME/db/data -Dweblogic.Domain=%DOMAIN_NAME%
-Dweblogic.Name=%SERVER_NAME% -Dbea.home=%BEA_HOME%
-Djava.security.policy==%WEBLOGIC_HOME%/lib/weblogic.policy
-Dcommerce.properties=PORTAL_HOME/weblogiccommerce.properties
weblogic.Server
```

Shutting Down a Server

To shut down the portalServer and portalDomain, use the `StopPortal` script, which is located in the WebLogic Portal installation directory.

To use the `StopPortal` script on UNIX, open a shell and enter the following command:

```
PORTAL_HOME/StopPortal.sh
```

To use the `StopPortal` script on Windows, enter the following command from a DOS prompt or equivalent command prompt:

```
PORTAL_HOME/StopPortal.bat
```

where `PORTAL_HOME` is the directory in which you installed WebLogic Portal.

Although it is possible to shut down the server by closing the shell that contains the process or by typing CTRL+C in the shell, such an abrupt action can cause transactions to be rolled back and any uncommitted session data to be lost. In addition, you can use the WebLogic Server Administration Console to shut down a WebLogic Portal instance.

Part II Deploying Your Business Data

CHAPTER

9 Configuring WebLogic Portal for Oracle Databases

WebLogic Portal includes a set of scripts that create Oracle tables and other data objects for use in any WebLogic Portal environment. Depending on the database environment that you are defining, you can choose to load sample data that supports the reference applications.

Usually, a development environment loads the sample data and a staging or production environment does not. In some cases, you may need to modify the default scripts so that they load non-operational sample data (such as data for the product catalog).

This section covers two types of Oracle configurations:

- Configuring for jDriver
- Configuring for Thin Driver (Available with Service Pack 2 Installation)

Configuring for jDriver

Step 1: Install the Oracle Client Software

Install the Oracle Client software on the same host as WebLogic Portal. Configure the Oracle Net Service Name to specify the target Oracle database. Then test the connection with the database user name and password that your database administrator assigned to you.

For information on the Oracle releases that WebLogic Portal supports for your platform type, refer to Supported Platforms in the *Installation Guide*.

Step 2: Create WebLogic Portal Tablespaces and a Schema-Owner User Account

To separate its data from the Oracle data dictionary and from data that other applications use, WebLogic Portal uses the following tablespaces:

- WLCS_DATA, which contains tables for WebLogic Portal
- WLCS_INDEX, which contains indexes for WebLogic Portal
- WLCS_EVENT_DATA, which contains tables for Behavior Tracking (WebLogic Portal needs this tablespace only if you use the Behavior Tracking feature. For information on setting up this tablespace, refer to *Guide to Events and Behavior Tracking*.)

A script named `create_tablespaces.sql` creates the `WLCS_DATA` and `WLCS_INDEX` tablespaces. It locates both table data and indexes in the `WLCS_DATA` tablespace. If possible, each tablespace should be placed on a separate physical disk drive. In Step 6: Rebuild Indexes, you move the indexes to the `WLCS_INDEX` tablespace.

By default, the schema-owner user account for WebLogic Portal is named `WEBLOGIC` and the tablespace is `WLCS_DATA` with a temporary tablespace named `TEMP`. The sample script named `create_users.sql` creates the `WEBLOGIC` user with default tablespaces and grants the `WEBLOGIC` user appropriate database permissions.

If you need to locate multiple instances of the WebLogic Portal schema on a single database, you must use a different schema-owner user account for each instance. For example, if you need to locate development and test database schemas on the same database instance, you could create two schema owner user accounts named `WL_DEV` and `WL_TEST`.

Creating WLCS_DATA and WLCS_INDEX

Note: In this document, `PORTAL_HOME` refers to the directory into which you installed WebLogic Portal.

To create the `WLCS_DATA` and `WLCS_INDEX` tablespaces, do the following (usually, a Database Administrator must complete these tasks):

1. From the host on which you installed the Oracle client and WebLogic Portal, open a UNIX shell or MS-DOS command prompt window and change directories to `PORTAL_HOME/db/oracle/817/admin`.

2. Make a backup copy of `create_tablespaces.sql`.

3. In a text editor, open `create_tablespaces.sql` and modify the pathnames for the `DATA_PATHNAME` and `INDEX_PATHNAME` variables to match your own local directory path structures. For example, on a UNIX system, if two disks are mounted as `/usr1` and `/usr2` and the Oracle SID is `PROD`, use the following pathnames:
   ```
   DEFINE DATA_PATHNAME=/usr1/oradata/PROD
   DEFINE INDEX_PATHNAME=/usr2/oradata/PROD
   ```

4. If you want to use a schema-owner user account other than the default `WEBLOGIC`, do the following:

   a. Make a backup copy of `create_users.sql`.

   b. In a text editor, open `create_users.sql` and change `WEBLOGIC` to the schema owner user account of your choice.

5. Enter the following command to start a SQL*Plus session:

   `sqlplus` *`username/password@net_service_name`*

   where *`username`* is `system` or a user with privileges to create tablespaces, *`password`* is the system (or other users) password, and *`net_service_name`* is the Net Service name that you defined for the Oracle database.

6. Enter the following to create tablespaces:

   `@create_tablespaces.sql`

   Output from executing `create_tablespaces.sql` will be written to a file named `CREATE_TABLESPACES.LOG`. Verify the results of running `create_tablespaces.sql` before you continue.

7. Enter the following to create a schema-owner user account:

   `@create_users.sql`

   Output from executing `create_users.sql` will be written to a file named `CREATE_USERS.LOG`. Verify the results of running `create_users.sql` before you continue.

8. Exit SQL*Plus.

Determining the Size of the WebLogic Portal Tablespaces

Monitoring the growth and usage of database resources and data files is a critical task that a Database Administrator must do on a regular basis. By default, the `WLCS_DATA` and `WLCS_INDEX` tablespaces are 100 Megabytes each. Any of the following factors might require you to adjust these default sizes:

- Whether you are working in a development, testing, staging, or production environment.
- The WebLogic Portal components (services) that your site uses. For example, a site that uses commerce, personalization, and portal services will probably require more disk space than a comparable site that uses only the personalization service.
- The number of users and groups for your application and the amount of data personalization data that the application generates and stores.
- Sizing for rollback segments, redo log files, archive log files and system tablespaces.

To determine the amount of disk space that you need in your production environment, a Database Administrator can analyze the usage of the database tables in the development environment. Based on the number of users, groups, personalization data and data from other services, the Database Administrator can project estimates for disk space requirements.

Step 3: Configure Properties and Environment Variables for Oracle

When you install WebLogic Portal, it is configured to support a Cloudscape demonstration database. This section describes how to modify the default Cloudscape properties to support an Oracle database.

To configure properties files and environment variables for Oracle, complete the following tasks in the order indicated below:

1. Start the WebLogic Server Administration Console
2. Configure JDBC Connection Pools for Oracle
3. Update Settings for the RDBMS Security Realm
4. Configure the JDBC Helper Service
5. Stop the Server
6. Update Environment Variables for the Server

Start the WebLogic Server Administration Console

To configure a WebLogic Portal server to support Oracle, start the server and access the WebLogic Server Administration Console for the server's domain.

For example, you want to configure `myServer` (which listens on port 7501). Start `myServer` on your local host and then enter the following URL in a browser:
`http://[host]:[port]/console`

For more information on starting the WebLogic Server Administration Console, refer to "WebLogic Server Administration Console" in the *WebLogic Portal Architectural Overview*.

Configure JDBC Connection Pools for Oracle

Connection pools provide ready-to-use pools of connections to your RDBMS. The application server creates the pools during server startup, thus eliminating the overhead of your enterprise application having to establish database connections for each transaction. For more information about connection pools, refer to “Overview of Connection Pools” in the *WebLogic Server Programming WebLogic JDBC* guide.

This section contains the following subsections:

- Configure commercePool for Oracle Databases
- Configure dataSyncPool for Oracle Databases

Configure commercePool for Oracle Databases

To configure `commercePool` for Oracle, do the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click commercePool.
3. On the General tab, enter the following values:

In this box...	**Enter this value for WebLogic jDriver**
URL	`jdbc:weblogic:oracle`
Driver Classname	`weblogic.jdbc.oci.Driver`

Properties	`user=@ORACLE_USER@<Return>`
	`weblogic.t3.waitForConnection=true<Return>`
	`weblogic.t3.waitSecondsForConnection=999999999999,web`
	`logic.jts.waitSecondsForConnectionSecs=999999999999,v`
	`erbose=false<Return>`
	`server=@ORACLE_NET_SERVICE_NAME@`
	
	Where <Return> denotes that you should hit the Return key after this line of text while entering properties in the "Properties" box. *`ORACLE_USER`* is the name of the Oracle user account (WEBLOGIC by default) and *`ORACLE_NET_SERVICE_NAME`* is the name of the Oracle net service or database SID.

5. Next to Password, click change.
6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.
7. To save your updated password, on the commercePool page, click Apply.

Configure dataSyncPool for Oracle Databases

Configure a JDBC connection pool named `dataSyncPool` by doing the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click dataSyncPool.
3. On the General tab, enter the following values:

In this box...	Enter this value for WebLogic jDriver
URL	`jdbc:weblogic:oracle`
Driver Classname	`weblogic.jdbc.oci.Driver`

Properties	`user=@ORACLE_USER@<Return>`
	`weblogic.t3.waitForConnection=true<Return>`
	`weblogic.t3.waitSecondsForConnection=999999999999,web`
	`logic.jts.waitSecondsForConnectionSecs=999999999999,v`
	`erbose=false<Return>`
	`server=@ORACLE_NET_SERVICE_NAME@`
	
	Where <Return> denotes that you should hit the Return key after this line of text while entering properties in the "Properties" box. *`ORACLE_USER`* is the name of the Oracle user account (WEBLOGIC by default) and *`ORACLE_NET_SERVICE_NAME`* is the name of the Oracle net service or database SID.

4. To save your entries, click Apply.
5. Next to Password, click change.

6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.

7. To save your updated password, on the dataSyncPool page, click Apply.

Update Settings for the RDBMS Security Realm

If you are using the RDBMS security realm, you must change the `RDBMSRealm` settings to match the database type that stores the user information. If you are using LDAP or some other security realm, you can ignore these settings.

To change these settings, do the following:

1. In the left pane of the WebLogic Server Administration Console, click Security → Realms → wlcsRealm.

   Note: If you named your RDBMS realm something other than wlcsRealm, click the realm that you created.

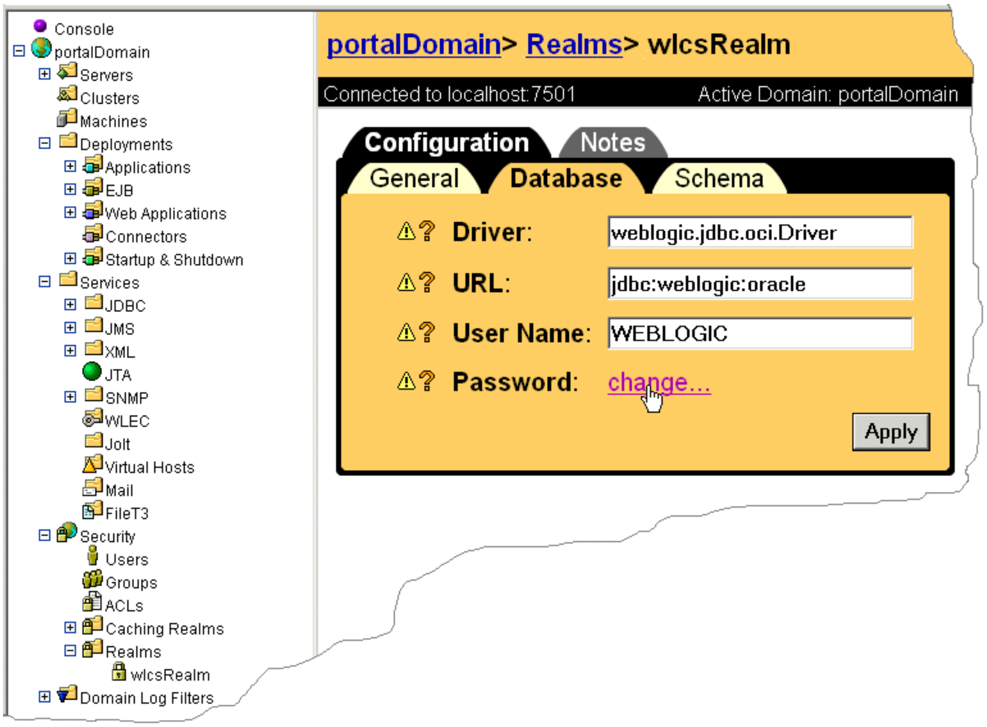
2. In the right pane, click the Database tab and enter the following values:

In this box...	Enter this value for WebLogic jDriver
URL	`jdbc:weblogic:oracle`
Driver Classname	`weblogic.jdbc.oci.Driver`
User Name	The name of the Oracle user account (WEBLOGIC by default).

3. To save your entries, click Apply.

4. Next to Password, click change. (See Figure 9-1.)

Figure 9-1 Modify Values on the Database Tab (jDriver Example)



Driver: weblogic.jdbc.oci.Driver
URL: jdbc:weblogic:oracle
User Name: WEBLOGIC
Password: change...

5. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Portal Administration Tools keeps this password in an encrypted format.

6. To save your password, on the wlcsRealm page, click Apply.

7. Click the Schema tab and enter the following properties:

```
user=@ORACLE_USER@<Enter>
weblogic.t3.waitForConnection=true<Enter>
weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.
waitSecondsForConnectionSecs=999999999999,verbose=false<Enter>
server=@ORACLE_NET_SERVICE_NAME@
```

where <Enter> denotes that you must press the Enter key after you type the line of text,
`@ORACLE_USER@` is the name of the Oracle user account (WEBLOGIC by default) and
`@ORACLE_NET_SERVICE_NAME@` is the name of the Oracle net service..

8. Click Apply.

Configure the JDBC Helper Service

The JDBC Helper Service enables services to explicitly establish a database connection and to coordinate the processing of CLOB data. To configure the JDBC Helper service for Oracle, **do the following for each deployed application that uses JDBC services**:

Note: Since the data synchronization framework uses the Personalization enterprise application, you will need to use the following procedure for the Personalization application as well as all other deployed applications.

1. In the left pane of the WebLogic Server Administration Console, click Deployment → Applications → *MyApplication* → Service Configuration → JDBC Helper Services.
2. In the right pane, on the Configuration tab, enter the following values:

In this box...	**Enter this value...**
Maximum Number of Retries	The maximum number of times the service attempts to connect to the database. A value of `-1` instructs the service to retry an unlimited number of times.
Maximum Wait Time	The number of milliseconds to wait for a database connection. When the time limit expires, the service attempts another connection, up to the maximum number of retries. A value of `-1` instructs the service to wait infinitely.
Delegate Class Name	`com.bea.p13n.util.jdbc.internal.OracleJDriverJdbcHelperDelegate`

3. To save your entries, click Apply.

4. Repeat these steps for each application that uses JDBC services and that you have deployed on this server instance.

Note: For information on setting the JDBC Helper Service for Sybase, DB2, and SQL Server 7, see step 2.4 of the section “Update Settings for the JDBC Helper Service” in the <PORTAL_HOME>/db/readme.html file. No Delegate Class Name is required for Cloudscape databases.

Configure the JDBC Helper Service when Migrating an Application

When migrating an application, including Petflow or the Avitek demonstration, the Service Configuration folder is not automatically created and therefore cannot be used to set up the Helper Service. To set up the Helper Service when migrating an application, do the following:

Note: This procedure configures the JDBC Helper Service for the jDriver for Oracle, it will not create a Service Configuration folder in the console.

1. In WLP_HOME/config/config.xml, enter the following within the `<application>` tag:

```
              <ApplicationConfiguration Name="<YourAppName>"
Targets="<yourappServername>"

              URI="META-INF/application-config.xml"/>
```

2. Create an `application-config.xml` file under WLP_HOME/applications/<YourAppName>/META-INF

3. In the Application-config.xml file input the following:

```
               <ApplicationConfiguration>
               <JdbcHelper

JdbcHelperDelegate="com.bea.p13n.util.jdbc.internal.OracleJDriver
JdbcHelperDelegate"
                     MaxRetries="-1"
                     MaxWaitTime="-1"
                     Name="JdbcHelper"

                   />
               </ApplicationConfiguration>
```

Stop the Server

Stop the server to complete the remainder of Step 3: Configure Properties and Environment Variables for Oracle.

For information on stopping the server, refer to “Shutting Down a Server” on page 8-9.

Update Environment Variables for the Server

To update environment variables, do the following:

1. Make a backup copy of `PORTAL_HOME\bin\win32\set-environment.bat` (Windows) or `PORTAL_HOME/bin/unix/set-environment.sh` (UNIX).
2. Open the file in a text editor and find the line `WLCS_ORACLE_HOME=@ORACLE_HOME@`

   The `WLCS_ORACLE_HOME` variable adds Oracle-client Java classes and libraries to the Java classpath and to the system path.
3. Substitute `@ORACLE_HOME@` with the absolute pathnames of the directory in which you installed the Oracle client software. For example, Listing 9-1 shows a Windows environment in which the Oracle client software is installed in `C:\oracle\ora81`.

Listing 9-1 Set @ORACLE_HOME@

```
SET PORTAL_HOME=c:\bea\Portal4.0
SET JDK_HOME=c:\bea\jdk1.3

SET WLCS_ORACLE_HOME=C:\oracle\ora81

SET BEA_HOME=c:\bea
SET WEBLOGIC_HOME=c:\bea\wlserver6.1
```

To specify the database driver, under the section titled `-------- Specify which database to use --------`, do one of the following:

- On UNIX, replace `# DATABASE=ORACLE_OCI` with **`SET DATABASE=ORACLE_OCI`**. Then deactivate the Cloudscape driver by placing # at the beginning of the Cloudscape line.
- On Windows, replace `REM DATABASE=ORACLE_OCI` with **`SET DATABASE=ORACLE_OCI`**. Then deactivate the Cloudscape driver by placing `REM` at the beginning of the Cloudscape line. (See Listing 9-2.)

Listing 9-2 Specify WebLogic jDriver

```
REM ----------- Specify which Database Driver to use -----------

REM SET DATABASE=CLOUDSCAPE

...

SET DATABASE=ORACLE_OCI
```

Step 4: Create the Database Schema for Oracle

WebLogic Portal includes a script, `create_all.bat` (`create_all.sh` for UNIX), that calls a series of other scripts to create the WebLogic Portal schema and install sample data. You can modify the script to create the database without loading sample data.

This step includes the following tasks:

- Set Variables in create_all and databaseload.properties
- Run create_all

Note: If you use the Event and Behavior Tracking Service, you must use a separate set of scripts to create the Event and Behavior Tracking schema objects. For more information, refer to “Scripts” under “Persisting Behavior Tracking Data” in the *Guide to Events and Behavior Tracking*.

Set Variables in create_all and databaseload.properties

To set variables in `create_all`, do the following:

1. Create a backup copy of `PORTAL_HOME/db/create_all.bat` (`create_all.sh` on UNIX).
2. Open the file in a text editor.
3. Find the following statements, which are near the beginning of the file:

```
REM To be set by user
    set SAMPLEDATA=Y
    set USER_ID=myUser
    set PASSWORD=myPassword
    set SERVER=myServer
REM End of set by user
```

4. Supply the following values for each statement:

   Note: `USER_ID` and `PASSWORD` are **case sensitive**. You must use the same case in `databaseload.properties` as you used when you created the database user (for example, `WEBLOGIC`, not `weblogic`).

For this statement...	Enter this value...
`SAMPLEDATA`	`Y` if you want to load sample catalog, customer, and user data. The reference applications use this data to demonstrate WebLogic Portal features. `N` if you do not want to load sample data. The reference applications do not function properly without this sample data. Do not enter `N` if you are setting up an environment in which you want to run the sample applications.

For this statement...	Enter this value...
`USER_ID`	The name of the Oracle user account (WEBLOGIC by default). This value is case sensitive and must be in the same case as the database user name.
`PASSWORD`	The password for the Oracle user account. This value is case sensitive and must be in the same case as the database user name.
`SERVER`	The net service name for the Oracle database.

5. Save `create_all`.

To set up `databaseload.properties`, do the following:

1. Create a backup copy of `PORTAL_HOME/db/databaseload.properties`.
2. Open `databaseload.properties` in a text editor.
3. Disable the Cloudscape statements by placing a pound character (#) at the beginning of each line:

   ```
   #------Cloudscape--------------------------------#
   #jdbcdriver=COM.cloudscape.core.JDBCDriver
   #connection=jdbc:cloudscape:Commerce
   #dblogin=none
   #dbpassword=none
   ```

4. Enable the statements for the Oracle driver by removing the pound character (#) from the beginning of each line. For example, the following lines specify the WebLogic jDriver for Oracle:

   ```
   #------Oracle OCI jDriver------------------------#
   jdbcdriver=weblogic.jdbc.oci.Driver
   connection=jdbc:weblogic:oracle:@ORACLE_NET_SERVICE_NAME@
   dblogin=@ORACLE_USER@
   dbpassword=@ORACLE_PASSWORD@
   ```

5. In the Oracle statements, replace `@ORACLE_USER@` with the name of the Oracle user account. Replace `@ORACLE_PASSWORD@` with the password for the user account and replace `@ORACLE_NET_SERVICE_NAME@` with the name of the net service that hosts the WebLogic Portal database.

6. Save `databaseload.properties`.

Note: You cannot use the Oracle driver when loading sample data via `create_all`. Instead, use jDriver for this step and in `set-environment` use `SET DATABASE=ORACLE_OCI` while running `create_all`. Refer to the Release Notes for more information.

Run create_all

Caution: Before it creates tables, `create_all.bat` runs SQL statements that drop any existing WebLogic Portal tables. If you run `create_all.bat` for a database that already contains WebLogic Portal data, you will lose any WebLogic Portal data that is in the database.

To create the WebLogic Portal schema, do the following:

1. Open a UNIX shell or MS-DOS command prompt window and change directories to the `PORTAL_HOME/db` directory.
2. Enter the following command:
   `PORTAL_HOME/db/create_all.bat` (`create_all.sh` on UNIX)

   The script logs its actions in several log files, which it locates in the `PORTAL_HOME/db` directory.
3. View the following log files to verify that there are no errors:
   - `drop_all.log`. If you run `create_all.bat` for a database that contains tables that are not part of the WebLogic Portal schema, the log file contains messages that state Cannot drop the table.... You can ignore these messages.
   - `create_all.log`
   - `insert_all.log`
   - `statistics.log`
   - `install_report.log`

For a description of the tables, indexes, and constraints, see the following topics:

- “The Personalization Server Schema” in the *Guide to Building Personalized Applications*

- “The Portal Management Database Schema” in the *Getting Started with Portals and Portlets*
- “Product Catalog Database Schema” in the *Guide to Building a Product Catalog*
- “Order Processing Database Schema” in the *Guide to Managing Purchases and Processing Orders*
- “Persisting Behavior Tracking Data” in the *Guide to Events and Behavior Tracking*
- “The Campaign Manager Database Schema” in the *Guide to Developing Campaign Infrastructure*

Step 5: Load Sample Data

You can load additional sample data that demonstrates ad placeholders and other Campaign services.

Note: You must load sample data in order to run the WebLogic Portal sample applications with Oracle. If you did not want to run the sample applications, then you can skip this step.

If you do not load the sample data and then try to bring up the portal tools (`http://host:7501/portalTools/index.jsp`) after starting the server, the following error message appears in the console:

```
<Error> <Webflow> <Error while parsing uri
/portalTools/application, path null, query string
namespace=admin_main - Webflow XML does not exist, is not
loaded properly, or you do not have a
configuration-error-page
defined.Exception[com.bea.p13n.appflow.exception.Configurat
ionException: The configuration-error-page node was not found
in the webflow xml file. for webapp [tools], namespace
[admin_main]. While trying to display CONFIGURATION ERROR:
[Exception[com.bea.p13n.appflow.exception.ConfigurationExce
ption: Bad Namespace - namespace [admin_main] is not
available for webflow execution. Make sure the
[admin_main.wf] file is deployed in webapp [tools].]],]
```

To load sample data, do the following:

1. Start the server that you want to use the sample data. For information on starting servers, refer to Chapter 8, “Starting and Shutting Down a Server.”.
2. Open a command shell and enter the following command:
   `PORTAL_HOME\bin\`*`platform-type`*`\loadSampleData.bat`
3. Shut down WebLogic Portal. For information on stopping the server, refer to “Shutting Down a Server” on page 8-9.

The `loadSampleData` script does the following:

- Synchronizes sample data with the reference applications.
- Calls `loaddocs.bat`, which loads documents that the p13nApp application displays.
- Calls `loadads.bat`, which loads ads to demonstrate ad placeholders in the wlcsApp reference application.

Step 6: Rebuild Indexes

The `create_all` script places the WebLogic Portal indexes in the default tablespace (`WLCS_DATA`). To locate and rebuild the indexes in the `WLCS_INDEX` tablespace, do the following:

1. From the host on which you installed the Oracle client and WebLogic Portal, open a UNIX shell or MS-DOS command prompt window and change directories to `PORTAL_HOME/db/oracle/817/admin`.
2. Enter the following command to start a SQL*Plus session:

   ```
   sqlplus username/password@net_service_name
   ```

   where *`username`* is the name of the Oracle user account (WEBLOGIC by default),
   `password` is the password for the Oracle user account, and
   `net_service_name` is the Net Service name that you defined for the Oracle database.
3. Enter the following command to rebuild indexes:
   `@rebuild_indexes.sql`

Configuring for Thin Driver

Thin Driver is certified with the Service Pack 2 Installation.

Step 1: Install the Oracle Client Software

Install the Oracle Client software on the same host as WebLogic Portal. Configure the Oracle Net Service Name to specify the target Oracle database. Then test the connection with the database username and password that your database administrator assigned to you.

For information on the Oracle releases that WebLogic Portal supports for your platform type, refer to Supported Platforms in the *Installation Guide*.

Step 2: Create WebLogic Portal Tablespaces and a Schema-Owner User Account

To separate its data from the Oracle data dictionary and from data that other applications use, WebLogic Portal uses the following tablespaces:

- WLCS_DATA, which contains tables for WebLogic Portal
- WLCS_INDEX, which contains indexes for WebLogic Portal
- WLCS_EVENT_DATA, which contains tables for Behavior Tracking (WebLogic Portal needs this tablespace only if you use the Behavior Tracking feature. For information on setting up this tablespace, refer to *Guide to Events and Behavior Tracking*.)

A script named `create_tablespaces.sql` creates the `WLCS_DATA` and `WLCS_INDEX` tablespaces. It locates both table data and indexes in the `WLCS_DATA` tablespace. If possible, each tablespace should be placed on a separate physical disk drive. In Step 6: Rebuild Indexes, you move the indexes to the `WLCS_INDEX` tablespace.

By default, the schema-owner user account for WebLogic Portal is named `WEBLOGIC` and the tablespace is `WLCS_DATA` with a temporary tablespace named `TEMP`. The sample script named `create_users.sql` creates the `WEBLOGIC` user with default tablespaces and grants the `WEBLOGIC` user appropriate database permissions.

If you need to locate multiple instances of the WebLogic Portal schema on a single database, you must use a different schema-owner user account for each instance. For example, if you need to locate development and test database schemas on the same database instance, you could create two schema owner user accounts named `WL_DEV` and `WL_TEST`.

Creating WLCS_DATA and WLCS_INDEX

Note: In this document, `PORTAL_HOME` refers to the directory into which you installed WebLogic Portal.

To create the `WLCS_DATA` and `WLCS_INDEX` tablespaces, do the following (usually, a Database Administrator must complete these tasks):

1. From the host on which you installed the Oracle client and WebLogic Portal, open a UNIX shell or MS-DOS command prompt window and change directories to `PORTAL_HOME/db/oracle/817/admin`.
2. Make a backup copy of `create_tablespaces.sql`.
3. In a text editor, open `create_tablespaces.sql` and modify the pathnames for the `DATA_PATHNAME` and `INDEX_PATHNAME` variables to match your own local directory path structures. For example, on a UNIX system, if two disks are mounted as `/usr1` and `/usr2` and the Oracle SID is `PROD`, use the following pathnames:
   ```
   DEFINE DATA_PATHNAME=/usr1/oradata/PROD
   DEFINE INDEX_PATHNAME=/usr2/oradata/PROD
   ```
4. If you want to use a schema-owner user account other than the default `WEBLOGIC`, do the following:
   a. Make a backup copy of `create_users.sql`.
   b. In a text editor, open `create_users.sql` and change `WEBLOGIC` to the schema owner user account of your choice.
5. Enter the following command to start a SQL*Plus session:

`sqlplus` *`username/password@net_service_name`*

where *`username`* is `system` or a user with privileges to create tablespaces, *`password`* is the system (or other users) password, and *`net_service_name`* is the Net Service name that you defined for the Oracle database.

6. Enter the following to create tablespaces:

   ```
   @create_tablespaces.sql
   ```

   Output from executing `create_tablespaces.sql` will be written to a file named `CREATE_TABLESPACES.LOG`. Verify the results of running `create_tablespaces.sql` before you continue.

7. Enter the following to create a schema-owner user account:

   ```
   @create_users.sql
   ```

   Output from executing `create_users.sql` will be written to a file named `CREATE_USERS.LOG`. Verify the results of running `create_users.sql` before you continue.

8. Exit SQL*Plus.

Determining the Size of the WebLogic Portal Tablespaces

Monitoring the growth and usage of database resources and data files is a critical task that a Database Administrator must do on a regular basis. By default, the `WLCS_DATA` and `WLCS_INDEX` tablespaces are 100 Megabytes each. Any of the following factors might require you to adjust these default sizes:

- Whether you are working in a development, testing, staging, or production environment.
- The WebLogic Portal components (services) that your site uses. For example, a site that uses commerce, personalization, and portal services will probably require more disk space than a comparable site that uses only the personalization service.
- The number of users and groups for your application and the amount of data personalization data that the application generates and stores.
- Sizing for rollback segments, redo log files, archive log files and system tablespaces.

To determine the amount of disk space that you need in your production environment, a Database Administrator can analyze the usage of the database tables in the development environment. Based on the number of users, groups, personalization data and data from other services, the Database Administrator can project estimates for disk space requirements.

Step 3: Configure Properties and Environment Variables for Oracle

When you install WebLogic Portal, it is configured to support a Cloudscape demonstration database. This section describes how to modify the default Cloudscape properties to support an Oracle database.

To configure properties files and environment variables for Oracle, complete the following tasks in the order indicated below:

1. Start the WebLogic Server Administration Console
2. Configure JDBC Connection Pools for Oracle
3. Update Settings for the RDBMS Security Realm
4. Configure the JDBC Helper Service
5. Stop the Server
6. Update Environment Variables for the Server

Start the WebLogic Server Administration Console

To configure a WebLogic Portal server to support Oracle, start the server and access the WebLogic Server Administration Console for the server's domain.

For example, you want to configure `myServer` (which listens on port 7501). Start `myServer` on your local host and then enter the following URL in a browser:
`http://[host]:[port]/console`

For more information on starting the WebLogic Server Administration Console, refer to "WebLogic Server Administration Console" in the *WebLogic Portal Architectural Overview*.

Configure JDBC Connection Pools for Oracle

Connection pools provide ready-to-use pools of connections to your RDBMS. The application server creates the pools during server startup, thus eliminating the overhead of your enterprise application having to establish database connections for each transaction. For more information about connection pools, refer to “Overview of Connection Pools” in the *WebLogic Server Programming WebLogic JDBC* guide.

This section contains the following subsections:

- Configure commercePool for Oracle Databases
- Configure dataSyncPool for Oracle Databases

Configure commercePool for Oracle Databases

To configure `commercePool` for Oracle, do the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click commercePool.
3. On the General tab, enter the following values:

In this box...	**Enter this value for WebLogic jDriver**
URL	`jdbc:oracle:thin:@@ORACLE_SERVER@:@oRACLE_PORT:@ORACLE_SID@`
Driver Classname	`oracle.jdbc.driver.OracleDriver`

In this box...	Enter this value for WebLogic jDriver
Properties	`user=@ORACLE_USER@<Return>` `weblogic.t3.waitForConnection=true<Return>` `weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.waitSecondsForConnectionSecs=999999999999,verbose=false<Return>` Where <Return> denotes that you should hit the Return key after this line of text while entering properties in the "Properties" box. *`ORACLE_USER`* is the name of the Oracle user account (WEBLOGIC by default) and *`ORACLE_NET_SERVICE_NAME`* is the name of the Oracle net service or database SID. **Note:** If you are changing the connection pool from the default Cloudscape to Oracle Thin, delete the last line in the Cloudscape setup: "server=none".

4. To save your entries, click Apply.
5. Next to Password, click change.
6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.
7. To save your updated password, on the commercePool page, click Apply.

Configure dataSyncPool for Oracle Databases

Configure a JDBC connection pool named `dataSyncPool` by doing the following:

1. In the left pane of the WebLogic Server Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click dataSyncPool.
3. On the General tab, enter the following values:

In this box...	Enter this value for WebLogic jDriver
URL	`jdbc:oracle:thin:@@ORACLE_SERVER@:@oRACLE_PORT:@ORACLE_SID@`
Driver Classname	`oracle.jdbc.driver.OracleDriver`

Properties	`user=@ORACLE_USER@<Return>`
	`weblogic.t3.waitForConnection=true<Return>`
	`weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.waitSecondsForConnectionSecs=999999999999,verbose=false<Return>`
	
	Where <Return> denotes that you should hit the Return key after this line of text while entering properties in the "Properties" box. *`ORACLE_USER`* is the name of the Oracle user account (WEBLOGIC by default) and *`ORACLE_NET_SERVICE_NAME`* is the name of the Oracle net service or database SID.
	
	Note: If you are changing the connection pool from the default Cloudscape to Oracle Thin, delete the last line in the Cloudscape setup: "server=none".

4. To save your entries, click Apply.
5. Next to Password, click change.
6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.
7. To save your updated password, on the dataSyncPool page, click Apply.

Update Settings for the RDBMS Security Realm

If you are using the RDBMS security realm, you must change the `RDBMSRealm` settings to match the database type that stores the user information. If you are using LDAP or some other security realm, you can ignore these settings.

To change these settings, do the following:

1. In the left pane of the WebLogic Server Administration Console, click Security → Realms → wlcsRealm.

   Note: If you named your RDBMS realm something other than wlcsRealm, click the realm that you created.

2. In the right pane, click the Database tab and enter the following values:

In this box...	Enter this value for WebLogic jDriver
URL	`jdbc:oracle:thin:@@ORACLE_SERVER@:@oRACLE_PORT:@ORACLE_SID@`
Driver Classname	`oracle.jdbc.driver.OracleDriver`
User Name	The name of the Oracle Thin user account (WEBLOGIC by default).

3. To save your entries, click Apply.

4. Next to Password, click change. (See Figure 9-1.)

5. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Portal Administration Tools keeps this password in an encrypted format.

6. To save your password, on the wlcsRealm page, click Apply.

7. Click the Schema tab and enter the following properties:

```
user=@ORACLE_USER@<Enter>
weblogic.t3.waitForConnection=true<Enter>
weblogic.t3.waitSecondsForConnection=999999999999,weblogic.jts.
waitSecondsForConnectionSecs=999999999999,verbose=false<Enter>
```

   where <Enter> denotes that you must press the Enter key after you type the line of text, *@ORACLE_USER@* is the name of the Oracle user account (WEBLOGIC by default)

8. Click Apply.

Configure the JDBC Helper Service

The JDBC Helper Service enables services to explicitly establish a database connection and to coordinate the processing of CLOB data. To configure the JDBC Helper service for Oracle, **do the following for each deployed application that uses JDBC services**:

Note: Since the data synchronization framework uses the Personalization enterprise application, you will need to use the following procedure for the Personalization application as well as all other deployed applications.

1. In the left pane of the WebLogic Server Administration Console, click Deployment → Applications → *MyApplication* → Service Configuration → JDBC Helper Services.
2. In the right pane, on the Configuration tab, enter the following values:

In this box...	Enter this value...
Maximum Number of Retries	The maximum number of times the service attempts to connect to the database. A value of `-1` instructs the service to retry an unlimited number of times.
Maximum Wait Time	The number of milliseconds to wait for a database connection. When the time limit expires, the service attempts another connection, up to the maximum number of retries. A value of `-1` instructs the service to wait infinitely.
Delegate Class Name	`com.bea.p13n.util.jdbc.internal.OracleThinJdbcHelperDelegate`

3. To save your entries, click Apply.
4. Repeat these steps for each application that uses JDBC services and that you have deployed on this server instance.

Note: For information on setting the JDBC Helper Service for Sybase, DB2, and SQL Server 7, see step 2.4 of the section "Update Settings for the JDBC Helper Service" in the <PORTAL_HOME>/db/readme.html file. No Delegate Class Name is required for Cloudscape databases.

Configure the JDBC Helper Service when Migrating an Application

When migrating an application, including Petflow or the Avitek demonstration, the Service Configuration folder is not automatically created and therefore cannot be used to set up the Helper Service. To set up the Helper Service when migrating an application, do the following:

Note: This procedure configures the JDBC Helper Service for the jDriver for Oracle, it will not create a Service Configuration folder in the console.

1. In WLP_HOME/config/config.xml, enter the following within the `<application>` tag:

```
              <ApplicationConfiguration Name="<YourAppName>"
Targets="<yourappServername>"

              URI="META-INF/application-config.xml"/>
```

2. Create an `application-config.xml` file under WLP_HOME/applications/<YourAppName>/META-INF

3. In the Application-config.xml file input the following:

```
              <ApplicationConfiguration>
              <JdbcHelper

JdbcHelperDelegate="com.bea.p13n.util.jdbc.internal.OracleThinJdb
cHelperDelegate"
                    MaxRetries="-1"
                    MaxWaitTime="-1"
                    Name="JdbcHelper"

                  />
              </ApplicationConfiguration>
```

Stop the Server

Stop the server to complete the remainder of Step 3: Configure Properties and Environment Variables for Oracle.

For information on stopping the server, refer to “Shutting Down a Server” on page 8-9.

Update Environment Variables for the Server

To update environment variables, do the following:

1. Make a backup copy of `PORTAL_HOME\bin\win32\set-environment.bat` (Windows) or `PORTAL_HOME/bin/unix/set-environment.sh` (UNIX).
2. Open the file in a text editor and find the line `WLCS_ORACLE_HOME=@ORACLE_HOME@`

   The `WLCS_ORACLE_HOME` variable adds Oracle-client Java classes and libraries to the Java classpath and to the system path.
3. Substitute `@ORACLE_HOME@` with the absolute pathnames of the directory in which you installed the Oracle client software. For example, Listing 9-1 shows a Windows environment in which the Oracle client software is installed in `C:\oracle\ora81`.

Listing 9-3 Set @ORACLE_HOME@

```
SET PORTAL_HOME=c:\bea\Portal4.0
SET JDK_HOME=c:\bea\jdk1.3

SET WLCS_ORACLE_HOME=C:\oracle\ora81

SET BEA_HOME=c:\bea
SET WEBLOGIC_HOME=c:\bea\wlserver6.1
```

To specify the database driver, under the section titled `-------- Specify which database to use --------`, do one of the following:

- On UNIX, replace `# DATABASE=ORACLE_OCI` with **`SET DATABASE=ORACLE_OCI`**. Then deactivate the Cloudscape driver by placing # at the beginning of the Cloudscape line.
- On Windows, replace `REM DATABASE=ORACLE_OCI` with **`SET DATABASE=ORACLE_OCI`**. Then deactivate the Cloudscape driver by placing `REM` at the beginning of the Cloudscape line. (See Listing 9-2.)

Listing 9-4 Specify WebLogic Thin Driver

```
REM ----------- Specify which Database Driver to use -----------

REM SET DATABASE=CLOUDSCAPE

...

SET DATABASE=ORACLE_THIN
```

Step 4: Create the Database Schema for Oracle

WebLogic Portal includes a script, `create_all.bat` (`create_all.sh` for UNIX), that calls a series of other scripts to create the WebLogic Portal schema and install sample data. You can modify the script to create the database without loading sample data.

This step includes the following tasks:

- Set Variables in create_all and databaseload.properties
- Run create_all

Note: If you use the Event and Behavior Tracking Service, you must use a separate set of scripts to create the Event and Behavior Tracking schema objects. For more information, refer to "Scripts" under "Persisting Behavior Tracking Data" in the *Guide to Events and Behavior Tracking*.

Set Variables in create_all and databaseload.properties

To set variables in `create_all`, do the following:

1. Create a backup copy of `PORTAL_HOME/db/create_all.bat` (`create_all.sh` on UNIX).
2. Open the file in a text editor.
3. Find the following statements, which are near the beginning of the file:

```
REM To be set by user
    set SAMPLEDATA=Y
    set USER_ID=myUser
    set PASSWORD=myPassword
    set SERVER=myServer
REM End of set by user
```

4. Supply the following values for each statement:

Note: `USER_ID` and `PASSWORD` are **case sensitive**. You must use the same case in `databaseload.properties` as you used when you created the database user (for example, `WEBLOGIC`, not `weblogic`).

For this statement...	Enter this value...
`SAMPLEDATA`	`Y` if you want to load sample catalog, customer, and user data. The reference applications use this data to demonstrate WebLogic Portal features. `N` if you do not want to load sample data. The reference applications do not function properly without this sample data. Do not enter `N` if you are setting up an environment in which you want to run the sample applications.

For this statement...	Enter this value...
`USER_ID`	The name of the Oracle user account (WEBLOGIC by default). This value is case sensitive and must be in the same case as the database user name.
`PASSWORD`	The password for the Oracle user account. This value is case sensitive and must be in the same case as the database user name.
`SERVER`	The net service name for the Oracle database.

5. Save `create_all`.

To set up `databaseload.properties`, do the following:

1. Create a backup copy of `PORTAL_HOME/db/databaseload.properties`.
2. Open `databaseload.properties` in a text editor.
3. Disable the Cloudscape statements by placing a pound character (#) at the beginning of each line:

```
#------Cloudscape--------------------------------#
#jdbcdriver=COM.cloudscape.core.JDBCDriver
#connection=jdbc:cloudscape:Commerce
#dblogin=none
#dbpassword=none
```

4. Enable the statements for the Oracle driver by removing the pound character (#) from the beginning of each line. For example, the following lines specify the WebLogic jDriver for Oracle:

Note: You must use jDriver to run `create_all` and `loadSampledata` and switch back the Thin Driver. (See the Release Notes document for more information.)

```
#------Oracle OCI jDriver------------------------#
jdbcdriver=oracle.jdbc.driver.OracleDriver
connection=jdbc:oracle:thin:@@ORACLE_SERVER@:@ORACLE_PORT@ORACL
E_SID@
dblogin=@ORACLE_USER@
dbpassword=@ORACLE_PASSWORD@
```

5. In the Oracle statements, replace `@ORACLE_USER@` with the name of the Oracle user account. Replace `@ORACLE_PASSWORD@` with the password for the user account and replace `@ORACLE_NET_SERVICE_NAME@` with the name of the net service that hosts the WebLogic Portal database.
6. Save `databaseload.properties`.

Note: You cannot use the Oracle Thin driver when loading sample data via `create_all.` Instead, use jDriver for this step and in `set-environment` use `SET DATABASE=ORACLE_OCI` while running `create_all`. Refer to the Release Notes for more information.

Run create_all

Caution: Before it creates tables, `create_all.bat` runs SQL statements that drop any existing WebLogic Portal tables. If you run `create_all.bat` for a database that already contains WebLogic Portal data, you will lose any WebLogic Portal data that is in the database.

To create the WebLogic Portal schema, do the following:

1. Open a UNIX shell or MS-DOS command prompt window and change directories to the `PORTAL_HOME/db` directory.
2. Enter the following command:
   `PORTAL_HOME/db/create_all.bat` (`create_all.sh` on UNIX)

   The script logs its actions in several log files, which it locates in the `PORTAL_HOME/db` directory.
3. View the following log files to verify that there are no errors:
   - `drop_all.log`. If you run `create_all.bat` for a database that contains tables that are not part of the WebLogic Portal schema, the log file contains messages that state Cannot drop the table.... You can ignore these messages.
   - `create_all.log`
   - `insert_all.log`
   - `statistics.log`
   - `install_report.log`

For a description of the tables, indexes, and constraints, see the following topics:

- "The Personalization Server Schema" in the *Guide to Building Personalized Applications*
- "The Portal Management Database Schema" in the *Getting Started with Portals and Portlets*
- "Product Catalog Database Schema" in the *Guide to Building a Product Catalog*
- "Order Processing Database Schema" in the *Guide to Managing Purchases and Processing Orders*
- "Persisting Behavior Tracking Data" in the *Guide to Events and Behavior Tracking*
- "The Campaign Manager Database Schema" in the *Guide to Developing Campaign Infrastructure*

Step 5: Load Sample Data

You can load additional sample data that demonstrates ad placeholders and other Campaign services.

Note: You must load sample data in order to run the WebLogic Portal sample applications with Oracle. If you did not want to run the sample applications, then you can skip this step.

If you do not load the sample data and then try to bring up the portal tools (`http://host:7501/portalTools/index.jsp`) after starting the server, the following error message appears in the console:

```
<Error> <Webflow> <Error while parsing uri
/portalTools/application, path null, query string
namespace=admin_main - Webflow XML does not exist, is not
loaded properly, or you do not have a
configuration-error-page
defined.Exception[com.bea.p13n.appflow.exception.Configurat
ionException: The configuration-error-page node was not found
in the webflow xml file. for webapp [tools], namespace
[admin_main]. While trying to display CONFIGURATION ERROR:
[Exception[com.bea.p13n.appflow.exception.ConfigurationExce
ption: Bad Namespace - namespace [admin_main] is not
available for webflow execution. Make sure the
[admin_main.wf] file is deployed in webapp [tools].]],]
```

To load sample data, do the following:

1. Start the server that you want to use the sample data. For information on starting servers, refer to Chapter 8, "Starting and Shutting Down a Server.".
2. Open a command shell and enter the following command:
   `PORTAL_HOME\bin\`*`platform-type`*`\loadSampleData.bat`
3. Shut down WebLogic Portal. For information on stopping the server, refer to "Shutting Down a Server" on page 8-9.

The `loadSampleData` script does the following:

- Synchronizes sample data with the reference applications.
- Calls `loaddocs.bat`, which loads documents that the p13nApp application displays.
- Calls `loadads.bat`, which loads ads to demonstrate ad placeholders in the wlcsApp reference application.

Step 6: Rebuild Indexes

The `create_all` script places the WebLogic Portal indexes in the default tablespace (`WLCS_DATA`). To locate and rebuild the indexes in the `WLCS_INDEX` tablespace, do the following:

1. From the host on which you installed the Oracle client and WebLogic Portal, open a UNIX shell or MS-DOS command prompt window and change directories to `PORTAL_HOME/db/oracle/817/admin`.
2. Enter the following command to start a SQL*Plus session:

   `sqlplus` *`username/password@net_service_name`*

   where *`username`* is the name of the Oracle user account (WEBLOGIC by default),
   `password` is the password for the Oracle user account, and
   `net_service_name` is the Net Service name that you defined for the Oracle database.
3. Enter the following command to rebuild indexes:
   `@rebuild_indexes.sql`

10 Configuring WebLogic Portal for Microsoft SQL Server Databases

WebLogic Portal includes a set of scripts that create Microsoft SQL Server tables and other data objects for use in any WebLogic Portal environment. The scripts run only in a Windows environment. If you use WebLogic Portal on a UNIX host with a Microsoft SQL Server database located on a Windows host, you must install the contents of the `PORTAL_HOME\db\sql_server\2000` directory and the `PORTAL_HOME\db\create_all.bat` script on a Windows machine with SQL Server database connectivity.

Depending on the database environment that you are defining, you can choose to load sample data that supports the reference applications. Usually, a development environment loads the sample data and a staging or production environment does not. In some cases, you may need to modify the default scripts so that they load non-operational sample data (such as data for the product catalog).

To use a Microsoft SQL Server database with a WebLogic Portal installation, complete the following tasks:

Step 1: Install the Client Software

Step 2: Create a User Database and Database Owner

Step 3: Configure Properties and Environment Variables for SQL Server

Step 4: Create the Schema Objects for MS SQL Server

Step 5: Load Additional Sample Data

Before you continue, follow the procedures in “Installing and Using WebLogic jDriver for Microsoft SQL Server” on the BEA WebLogic Server documentation site.

Step 1: Install the Client Software

On the WebLogic Portal host, use the SQL Server installation program to install the SQL Server client software and configure it to access your SQL Server Database.

For information on the SQL Server release that WebLogic Portal supports, refer to Supported Platforms in the *Installation Guide*.

After you install the client software, complete the following tasks:

- Configure Security Authentication
- Create and Test the ODBC Data Source

Configure Security Authentication

Verify that the security authentication settings for your SQL Server are set to “SQL Server and Windows.” You cannot create the WebLogic Portal database if your SQL Server is configured for “Windows Only” authentication. To configure authentication settings, do the following:

1. From the SQL Server Enterprise Manager, right click your SQL Server.
2. From the shortcut menu, select Properties.
3. In the Properties window, on the Security Tab, select SQL Server and Windows.

Create and Test the ODBC Data Source

Before you can run `create_database.sql`, verify that an ODBC Data Source for Microsoft SQL Server is installed on the WebLogic Portal host.

Then use the `osql` utility to make sure that you can connect to the SQL Server database from the WebLogic Portal host.

Step 2: Create a User Database and Database Owner

You must create a user database and database owner exclusively for the WebLogic Portal schema. If you need to locate multiple WebLogic Portal databases on same database server, you must create a separate database and database owner combination for each database.

Note: We recommend that you locate the SQL Server database on a host other than the WebLogic Portal host.

WebLogic Portal includes a sample script, `create_database.sql`, for creating a user database named WLCS with an owner named WEBLOGIC.

To create a user database and database owner, complete the following tasks:

- Modify create_database.sql
- Run create_database.sql
- Change the Password

Modify create_database.sql

You must modify `create_database.sql` to substitute pathnames that match your environment. We recommend that you place the database data file and the log file on separate physical disks and away from any system database (i.e. master, msdb, tempdb) data or log files. The script provides initial sizes for these files, and you can adjust the values depending on your disk-space requirements and system usage.

To modify the script, do the following:

Note: If you use WebLogic Portal from a Unix host with a Microsoft SQL Server database located on a Windows host, edit the `create_database.sql` that you copied to the Windows machine.

1. Create a backup copy of
   `PORTAL_HOME\db\sql_server\2000\admin\create_database.sql`

   Note: In this document, `PORTAL_HOME` refers to the directory into which you installed WebLogic Portal.

2. Open `create_database.sql` in a text editor.

3. To specify a pathname for the data file, under the `NAME=WLCS_DATA` statement, change `FILENAME='D:\DATAFILE\WLCS_DATA.mdf'` to
   `FILENAME = '`*`my-pathname\my-data-file.mdf`*`'`
   where *`my-pathname\my-data-file.mdf`* is the path and filename you want to use.

   If you create more than one WebLogic Portal database on the same SQL Server host, each data file must have a unique name.

4. To specify a pathname for the log file, under the `NAME=WLCS_LOG` statement, change `FILENAME = 'E:\LOGFILE\WLCS_LOG.ldf'`to
   `FILENAME = '`*`my-pathname\my-log-file.mdf`*`'`
   where *`my-pathname\my-log-file.mdf`* is the path and filename you want to use.

   If you create more than one WebLogic Portal database on the same SQL Server host, each log file must have a unique name.

5. (optional) To change the name of the database-owner user ID, substitute all instances of `WEBLOGIC` with your database-owner user ID. For example,
   ```
   go
   exec sp_addlogin 'my-database-owner', 'PASSWORD', 'WLCS',
   ```

```
'us_english'
go
use WLCS
go
sp_changedbowner my-database-owner
go
```

where *my-database-owner* is the user ID of your database owner.

6. (optional) To change the name of the database, substitute all instances of `WLCS` with the name of your database. In addition, you can update `WLCS_DATA` and `WLCS_LOG` to reflect the new name of your database. For example,
   use master

```
go
create database my-database-name
ON
(NAME = my-database-name_DATA,
FILENAME = 'D:\DATAFILE\WLCS_DATA.mdf',
SIZE = 60)
LOG ON

(NAME = my-database-name_LOG,
FILENAME = 'E:\LOGFILE\WLCS_LOG.ldf',
SIZE = 20)

go
checkpoint
go

-- *** Create a WEBLOGIC login and user in the WLCS database
use master
go
exec sp_addlogin 'WEBLOGIC', 'PASSWORD', 'my-database-name',
'us_english'
go
use my-database-name
go
sp_changedbowner WEBLOGIC
go
```

where *my-database-name* is the new name of your database.

7. Save your modifications to `create_database.sql`.

   Note: If you change the names of the database owner (`WEBLOGIC` by default) or the database (`WLCS` by default), you must review each script for the default names and modify them to match your new names.

Run create_database.sql

To run the `create_database.sql` script, do the following:

1. Open a MS-DOS Command Prompt window and cd to the following directory:

   ```
   PORTAL_HOME\db\sql_server\2000\admin
   ```

2. Enter the following command:

   ```
   osql -U sa -S SERVER -e -icreate_database.sql
   -ocreate_database.log
   ```

   where *`SERVER`* is the name of your SQL Server database server host.

3. When prompted, enter the password for the System Administrator (sa).

   The command logs on to the database server host as System Administrator, runs the `create_database.sql` script, and directs the output to a file named `.\create_database.log`.

4. View `create_database.log` to verify that there are no errors.

Change the Password

The `create_database.sql` script creates a generic password named `password`.

We recommend that you change the database-owner password to a new password of your choice.

To change the password, do the following from a DOS prompt:

1. Enter the following command:

   ```
   osql -U WEBLOGIC -P PASSWORD -S server-name
   ```

   where *`WEBLOGIC`* is the user name for the database owner and *`server-name`* is the name of your SQL Server database server host.

2. Enter the following command:

   ```
   1> sp_password PASSWORD,newpassword
   ```

   where *`newpassword`* is your new password.

3. Enter the following command:

   ```
   2> go
   ```

osql sets the new password and returns the following message:

```
Password correctly set.
(return status = 0)
1>
```

Step 3: Configure Properties and Environment Variables for SQL Server

When you install WebLogic Portal, it is configured to support a Cloudscape demonstration database. This section describes how to modify the default Cloudscape properties to support a SQL Server database.

To configure properties and environment variables for SQL Server, complete the following tasks in the order indicated below:

1. Start the Administration Console
2. Set Up JDBC Connection Pools and Data Sources
3. Update Settings for the RDBMS Security Realm
4. Configure the JDBC Helper Service
5. Stop the Server
6. Update Environment Variables for the Server

Start the Administration Console

To configure a server to support SQL Server, start the server that you want to configure and access the Administration Console for the server's domain.

For example, you want to configure `myServer` (which listens on port 7501). Start `myServer` on your local host and then enter the following URL in a browser:
`http://localhost:7501/console`

Set Up JDBC Connection Pools and Data Sources

Connection pools provide ready-to-use pools of connections to your RDBMS. The application server creates the pools during server startup, thus eliminating the overhead of your enterprise application having to establish database connections for each transaction. For more information about connection pools, refer to “Overview of Connection Pools” in the *WebLogic Server Programming WebLogic JDBC* guide.

A DataSource is an interface between your enterprise application and a connection pool. For more information, refer to “Overview of DataSources” in the *WebLogic Server Programming WebLogic JDBC* guide.

- Configure commercePool for SQL Server Databases
- Configure dataSyncPool for SQL Server Databases

Configure commercePool for SQL Server Databases

To configure `commercePool` for SQL Server, do the following:

1. In the left pane of the Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click commercePool.
3. On the General tab, enter the following values (See Figure 10-1):

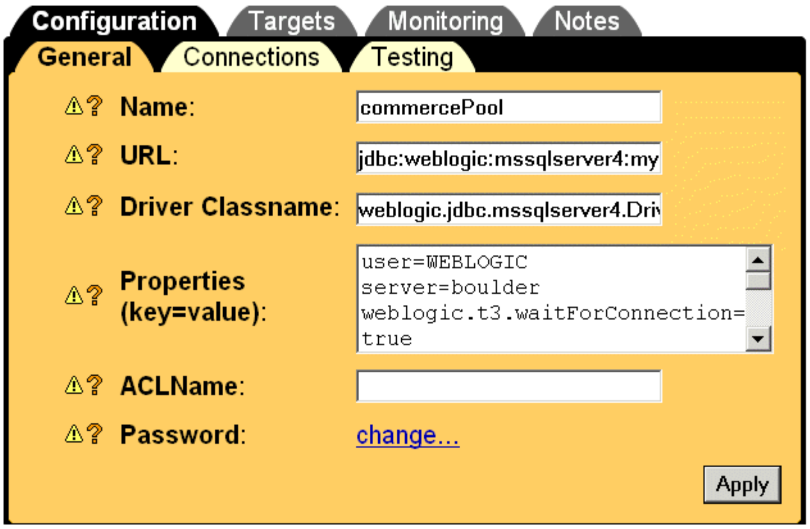
In this box...	**Enter this value...**
URL	`jdbc:weblogic:mssqlserver4:`*`yourServerName`*:*`port-number`* Where *`yourServerName`* is the name of the SQL Server database server that hosts the WebLogic Portal database and *`port-number`* is port that the database server is configured to listen on.
Driver Classname	`weblogic.jdbc.mssqlserver4.Driver`

In this box...	Enter this value...

Properties	`user=`*`@SQL_SERVER_USER@`*
	`server=`*`@SQL_SERVER_NAME@`*
	`weblogic.t3.waitForConnection=true`
	`weblogic.t3.waitSecondsForConnection=999999999`
	`999`
	`weblogic.jts.waitSecondsForConnectionSecs=9999`
	`99999999`
	`verbose=false`
	
	where *`@SQL_SERVER_USER@`* is the name of the database owner and
	`@SQL_SERVER_NAME@` is the name of the SQL Server database server that hosts the WebLogic Portal database.

ACLName	no value

Figure 10-1 The General Tab for JDBC Connection Pools



Name: commercePool

URL: jdbc:weblogic:mssqlserver4:my

Driver Classname: weblogic.jdbc.mssqlserver4.Dri

Properties (key=value):

```
user=WEBLOGIC
server=boulder
weblogic.t3.waitForConnection=
true
```

ACLName:

Password: change...

4. Click Apply.

5. Next to Password, click change.

6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.

7. To save your updated password, on the commercePool page, click Apply.

Configure dataSyncPool for SQL Server Databases

To configure `dataSyncPool` for SQL Server, do the following:

1. In the left pane of the Administration Console, click Services → JDBC → Connection Pools.
2. Under the Connection Pools folder, click dataSyncPool.
3. On the General tab, enter the following values:

In this box...	**Enter this value...**
URL	`jdbc:weblogic:mssqlserver4:`*`yourServerName`*:*`port-number`* Where *`yourServerName`* is the name of the SQL Server database server that hosts the WebLogic Portal database and *`port-number`* is port that the database server is configured to listen on.
Driver Classname	`weblogic.jdbc.mssqlserver4.Driver`

Properties	`user=`*`@SQL_SERVER_USER@`*
	`server=`*`@SQL_SERVER_NAME@`*
	`weblogic.t3.waitForConnection=true`
	`weblogic.t3.waitSecondsForConnection=999999999999`
	`weblogic.jts.waitSecondsForConnectionSecs=999999999999`
	`verbose=false`
	where *`@SQL_SERVER_USER@`* is the name of the database owner and *`@SQL_SERVER_NAME@`* is the name of the SQL Server database server that hosts the WebLogic Portal database.

ACLName	no value

4. Click Apply.
5. Next to Password, click change.
6. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Server Administration Console keeps this password in an encrypted format.
7. To save your updated password, on the dataSyncPool page, click Apply.

Update Settings for the RDBMS Security Realm

If you are using the RDBMS security realm, you must change the `RDBMSRealm` settings to match the database type that stores the user information. If you are using LDAP or some other security realm, you can ignore these settings.

To change these settings, do the following:

1. In the left pane of the WebLogic Server Administration Console, click Security → Realms → wlcsRealm.

   Note: If you named your RDBMS realm something other than wlcsRealm, click the realm that you created.
2. In the right pane, click the Database tab and enter the following values:

In this box...	Enter this value...
Driver	`weblogic.jdbc.mssqlserver4.Driver`
URL	`jdbc:weblogic:mssqlserver4:`*`yourServerName`*:*`port-number`* Where *`yourServerName`* is the name of the SQL Server database server that hosts the WebLogic Portal database and *`port-number`* is port that the database server is configured to listen on.
User Name	The name of the database owner.

3. To save your entries, click Apply.

4. Next to Password, click change.
5. On the Change Password page, enter and confirm the password of the user account. Then click Apply.

   The WebLogic Portal Administration Tools keeps this password in an encrypted format.
6. To save your password, on the wlcsRealm page, click Apply.
7. Click the Schema tab and enter the following properties:

```
weblogic.allow.reserve.weblogic.jdbc.connectionPool.commercePool=wlcs
weblogic.allow.reset.weblogic.jdbc.connectionPool.commercePool=wlcs
weblogic.allow.shrink.weblogic.jdbc.connectionPool.commercePool=wlcs
server=@SQL_SERVER_NAME@
```

   where *`@SQL_SERVER_NAME@`* is the name of the SQL Server database server that hosts the WebLogic Portal database.
8. Click Apply.

Configure the JDBC Helper Service

The JDBC Helper Service enables services to explicitly establish a database connection and to coordinate the processing of CLOB data. To configure the JDBC Helper service for SQL Server, **do the following for each application that you have deployed**:

Note: Since the data synchronization framework uses the Personalization enterprise application, you will need to use the following procedure for the Personalization application as well as all other deployed applications.

1. In the left pane of the WebLogic Server Administration Console, click Deployments → *myApplication* → Service Configuration → JDBC Helper Service.

2. In the right pane, on the Configuration tab, enter the following values:

In this box...	Enter this value...
Maximum Number of Retries	The maximum number of times the service attempts to connect to the database. A value of `-1` instructs the service to retry an unlimited number of times.
Maximum Wait Time	The number of milliseconds to wait for a database connection. When the time limit expires, the service attempts another connection, up to the maximum number of retries. A value of `-1` instructs the service to wait infinitely.
Delegate Class Name	`com.bea.p13n.util.jdbc.internal.GenericJdbcHelperDelegate`

3. To save your entries, click Apply.
4. Repeat these steps for each application that you have deployed in this server instance.

Stop the Server

Stop the server to complete the remainder of Step 3: Configure Properties and Environment Variables for SQL Server.

For information on stopping the server, refer to “Shutting Down a Server” on page 8-9.

Update Environment Variables for the Server

To update environment variables, do the following on your WebLogic Portal host:

1. In a text editor, open `PORTAL_HOME\bin\`*`platform-type`*`\set-environment.bat` (`set-environment.sh` on UNIX)

2. Under the section titled
   `-------- Specify which database to use --------,`
   activate the line that sets the `DATABASE` variable to SQL Server and deactivate any other line. For example, Listing 10-1 shows a Windows environment that uses SQL Server.

Listing 10-1 Specify the Database (Example for a Windows Environment)

```
REM ----------- Specify which database to use -----------

REM SET DATABASE=CLOUDSCAPE
REM SET DATABASE=ORACLE
REM SET DATABASE=ORACLE_OCI_815

SET DATABASE=MSSQL
```

Step 4: Create the Schema Objects for MS SQL Server

WebLogic Portal includes a script, `create_all.bat`, that calls a series of other scripts to create the WebLogic Portal schema and loads sample data. You can modify the script to create the database without loading sample data.

This step includes the following tasks:

- Set Variables in databaseload.properties
- Set Variables in create_all.bat
- Run create_all.bat

Set Variables in databaseload.properties

To set up `databaseload.properties`, do the following on your WebLogic Portal host:

Note: If you use WebLogic Portal from a Unix host, edit the `databaseload.properties` that is located on the UNIX host.

1. Create a backup copy of `PORTAL_HOME\db\databaseload.properties`.
2. Open `databaseload.properties` in a text editor.
3. Disable the Cloudscape statements by placing a pound character (#) at the beginning of each line:

```
#------Cloudscape---------------------------------#
#jdbcdriver=COM.cloudscape.core.JDBCDriver
#connection=jdbc:cloudscape:Commerce
#dblogin=none
#dbpassword=none
```

4. Enable the statements for SQL Server by removing the pound character (#) from the beginning of each line:

```
#------MS SQL Server-----------------------------#

jdbcdriver=weblogic.jdbc.mssqlserver4.Driver
connection=jdbc:weblogic:mssqlserver4:@MSSQL_SERVER@:@MSSQL_PORT@
dblogin=@MSSQL_USER@
dbpassword=@MSSQL_PASSWORD@
```

5. Replace `@MSSQL_SERVER@:@MSSQL_PORT@` with the name and listen port for your SQL Server.

   Replace `@MSSQL_USER@` with the name of the database owner and `@MSSQL_PASSWORD@` with the database owner's password.

   The user name and password must match the user name and password you used in "Set Up JDBC Connection Pools and Data Sources" on page 10-9 and "Update Settings for the RDBMS Security Realm" on page 10-12.

6. Save `databaseload.properties`.

Set Variables in create_all.bat

To set variables in `create_all.bat`, do the following:

Note: If you use WebLogic Portal from a Unix host with a Microsoft SQL Server database located on a Windows host, edit the `create_all.bat` that you copied to the Windows machine.

1. Create a backup copy of `PORTAL_HOME\db\create_all.bat`.

2. Open `create_all.bat` in a text editor.

3. Find the following statements, which are near the beginning of the file:

```
REM To be set by user
    set SAMPLEDATA=Y
    set USER_ID=myUser
    set PASSWORD=myPassord
    set SERVER=myServer
REM End of set by user
```

4. Supply the following values for each statement:

For this statement...	Enter this value...
SAMPLEDATA	Y if you want to load sample catalog, customer, and user data. The reference applications use this data to demonstrate WebLogic Portal features. N if you do not want to load sample data. The reference applications do not function properly without this sample data. Do not enter N if you are setting up an environment in which you want to run the sample applications.
USER_ID	The name of the SQL Server database owner.
PASSWORD	The database owner's password.
SERVER	The name of the SQL Server database server that hosts the WebLogic Portal database.

5. In `create_all.bat`, find the section that is illustrated below. Replace the `@WL_PORTAL_HOME@` variable with the absolute pathname of the directory in which you installed WebLogic Portal. Remove the `echo` statements and `EXIT` command from the code.

```
echo
****************************************************************

echo    Please replace the reference of @WL_PORTAL_HOME@
echo    with the actual WebLogic Portal path
        ( e.g.,C:\BEA\WLPORTAL4.0 )

echo   Once changed, be sure and remove these echo statements
        and the
echo    EXIT command from the batch file.
echo
****************************************************************

    EXIT

CALL @WL_PORTAL_HOME@\bin\win32\set-environment.bat
```

6. If you use WebLogic Portal from a Unix host, in `create_all.bat`, set the `MSSQL_UNIX` switch to `Y` (it is set to `N` by default). For example:

```
REM To be set by user for Microsoft SQL Server
REM set MSSQL_VERSION=7
set MSSQL_VERSION=2000

REM set MSSQL_UNIX=N
set MSSQL_UNIX=Y
```

7. Save `create_all.bat`.

Run create_all.bat

Caution: Before it creates tables, `create_all.bat` runs SQL statements that drop any existing WebLogic Portal tables. If you run `create_all.bat` for a database that already contains WebLogic Portal data, you will lose any WebLogic Portal data that is in the database.

To run this script, do the following:

1. Open a MS-DOS Command Prompt window and cd to the following directory:

   ```
   PORTAL_HOME\db\
   ```

2. Enter the following command:

   ```
   create_all.bat
   ```

   The script creates log files under `PORTAL_HOME\db`.

   The script logs its actions in several log files, which it locates in the `PORTAL_HOME\db` directory.

3. View the following log files to verify that there are no errors:
   - `drop_all.log`. If you run `create_all.bat` for a database that contains tables that are not part of the WebLogic Portal schema, the log file contains messages that state Cannot drop the table.... You can ignore these messages.
   - `create_all.log`
   - `insert_all.log`
   - `statistics.log`
   - `install_report.log`
4. If you use WebLogic Portal from a Unix host, run the following scripts from a Unix shell to insert both required and sample database bootstrap into the database tables:
   - `PORTAL_HOME/db/loadbootstrap.sh`
   - `PORTAL_HOME/db/loadsample.sh`

For a description of the tables, indexes, and constraints, see the following topics:

- "The Personalization Server Schema" in the *Guide to Building Personalized Applications*
- "The Portal Management Database Schema" in the *Guide to Developing and Managing Portals*
- "Product Catalog Database Schema" in the *Guide to Building a Product Catalog*
- "Order Processing Database Schema" in the *Guide to Managing Purchases and Processing Orders*
- "Persisting Behavior Tracking Data" in the *Guide to Events and Behavior Tracking*
- "The Campaign Manager Database Schema" in the *Guide to Developing Campaign Infrastructure*

Step 5: Load Additional Sample Data

You can load additional sample data that demonstrates ad placeholders and other Campaign services. To load additional sample data, do the following from the WebLogic Portal host:

1. Start the server that you want to use the sample data. For information on starting servers, refer to Chapter 8, “Starting and Shutting Down a Server.”.
2. Open a command shell and enter the following command:
   `PORTAL_HOME\bin\`*`platform-type`*`\loadSampleData.bat`
   (`loadSampleData.sh` on UNIX)
3. Shut down WebLogic Portal. For information on stopping the server, refer to “Shutting Down a Server” on page 8-9.

The `loadSampleData` script does the following:

- Synchronizes sample data with the p13nApp application and the wlcsApp application.
- Calls `loaddocs.bat`, which loads documents that the p13nApp application displays.
- Calls `loadads.bat`, which loads ads to demonstrate ad placeholders in the wlcsApp reference application.

Index