BEA Campaign Manager for WebLogic
BEA WebLogic Commerce Server
BEA WebLogic Personalization Server

Deployment Guide

BEA Campaign Manager for WebLogic 1.1
BEA WebLogic Personalization Server 3.5
BEA WebLogic Commerce Server 3.5
Document Edition 3.5.2
June 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Deployment Guide

Document Edition	Date	Software Version
3.5.2	June 2001	BEA Campaign Manager for WebLogic 1.1 BEA WebLogic Commerce Server 3.5 BEA WebLogic Personalization Server 3.5

Contents

2. The Reference Domain

3. Deploying Your Enterprise Application

4. Starting and Shutting Down the Server

Part II. Deploying the Data Repository

5. Setting Up Oracle for New Installations

6. Migrating Oracle Data Objects

Index

About This Document

When you install BEA WebLogic Commerce Server™, BEA WebLogic Personalization Server™, and BEA Campaign Manager for WebLogic™, it is configured to use a Cloudscape database and the BulkLoader content management system. When you are ready to configure Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server for developing, testing, and publishing your own e-commerce Web site and portals, follow the guidelines and procedures in this document.

This document includes the following topics:

- Chapter 1, "The Server Configuration," which describes reviewing default settings used to deploy Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.
- Chapter 2, "The Reference Domain," which describes reviewing default settings for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server domain and enterprise application.
- Chapter 3, "Deploying Your Enterprise Application," which describes deploying the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server enterprise application.
- Chapter 4, "Starting and Shutting Down the Server," which describes configuration and startup files for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.
- Chapter 5, "Setting Up Oracle for New Installations," which describes creating the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server database schemas for Oracle databases.
- Chapter 6, "Migrating Oracle Data Objects," which describes exporting data from one environment to the next (for example, from a development environment to a staging environment).

What You Need to Know

This document is intended mainly for Web site administrators and database administrators who configure properties for WebLogic Server and Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. It assumes a familiarity with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server documentation Home page on the e-docs Web site (and also on the documentation CD). 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server:

- *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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server **Product Version: 3.5** release.

If you have any questions about this version of BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, or if you have problems installing and running BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, 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.
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`** `( )`

Convention	Item
monospace italic text	Identifies variables in code. *Example*: `String expr`
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Part I Deploying the Server and Its Enterprise Application

1 The Server Configuration

When you install Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, the installation program sets up a fully deployed server with sample Web applications and supporting EJBs. The Web applications and EJBs are organized according to the J2EE specification for *enterprise applications*.

Before you modify the default configuration for your development or production environment, we recommend that you read the following sections in this topic and Chapter 2, "The Reference Domain," which describe the default configuration and indicate the properties that you can modify:

- The Directory Structure for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server
- Viewing and Modifying Properties in the WebLogic Server Administration Console
- Viewing and Modifying Properties in weblogiccommerce.properties

The Directory Structure for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server

By default, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server installs into the BEA home directory. The root of the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server directory tree contains the following (see Figure 1-1):

- `pipeline.properties`, `webflow.properties`, and `weblogiccommerce.properties`, which set run-time properties for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. We recommend that you make backup copies or place these files under source control before modifying these files for your specific configuration.
- `StartCommerce` and `StopCommerce`, which are scripts that start and stop Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. `StopCommerce` also stops some processes that support third-party integrations. For more information about these command, refer to Chapter 4, "Starting and Shutting Down the Server."
- `bin`, whose subdirectories contain platform-specific scripts for setting up your environment. The `set-environment` file in the platform-specific subdirectory sets all environment variables required for the run-time environment. For more information, see "Setting Environment Variables" on page 4-4.
- `db`, which contains the Cloudscape database for demonstration purposes only. We do not support using this database in a development or production environment. To see a list of databases that we do 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.

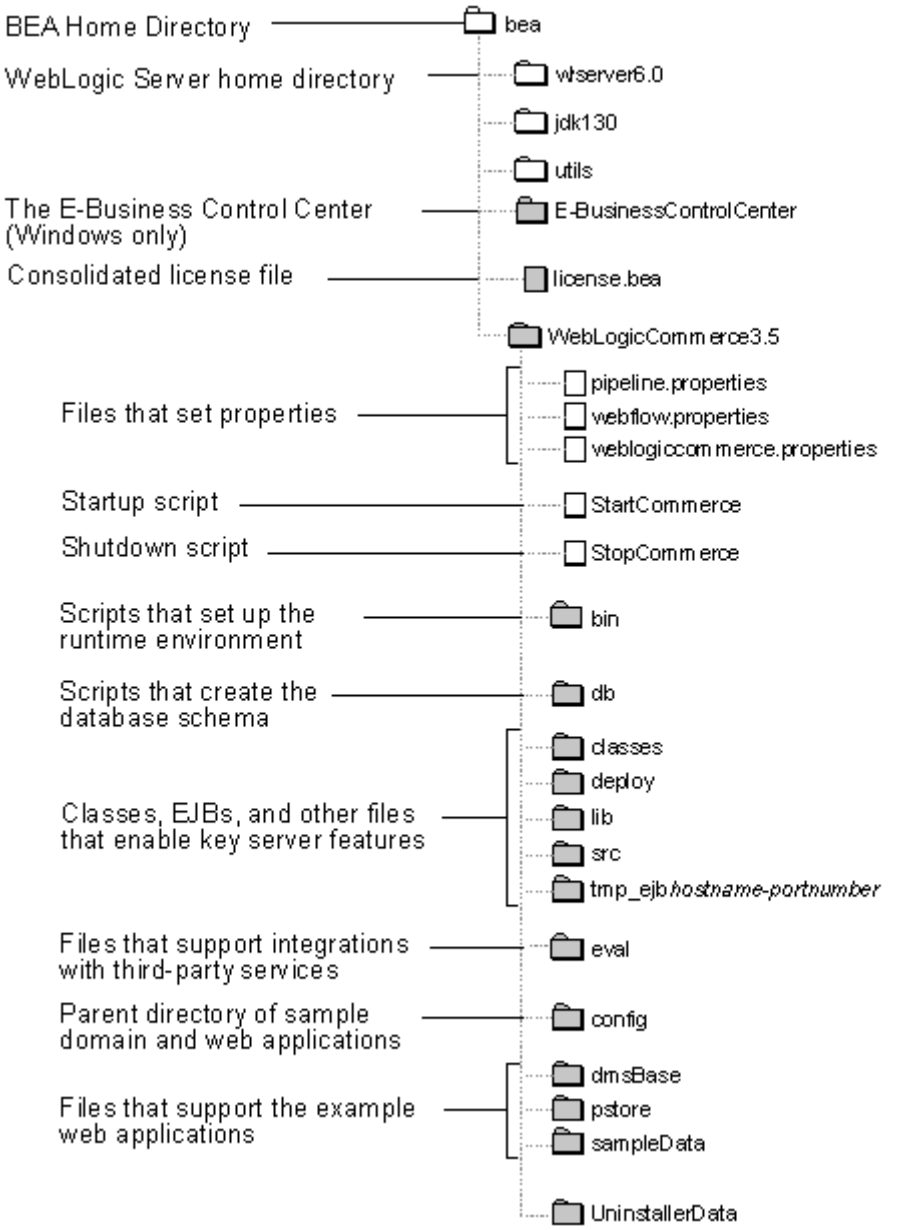
- `classes`, `deploy`, `lib`, `src`, which contain Java source files, compiled classes, EJBs, JARs, DTDs and other files that extend WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server features. The `classes` directory and a set of JAR files under the `lib` directory must be named in the `CLASSPATH` environment variable. (For more information, refer to "Add Directories to CLASSPATH" on page 4-6.) Do not modify any of these files and directories.
- `eval`, which contains files to support integration with third-party services, such as sales tax calculations and credit card validation. If you use these services, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server must have access to these files, though they do not need to be in the current directory tree. If you change the location of these files, you must modify the environment variables that the StartCommerce startup script sets. For more information, refer to "Setting Environment Variables" on page 4-4.
- `config`, which is the parent directory for the sample Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server domain. For more information, see Chapter 2, "The Reference Domain."
- `dmsBase`, `pstore`, and `sampleData`, which contain files to support the sample web applications. These directories need to be available only to those environments from which you want to access the sample portal web application.
- `UninstallerData`, which contains files for uninstalling Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.

Additional Files and Directories

On Windows platforms, you can install the BEA E-Business Control Center, a graphical user interface (GUI) for managing Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server features. By default, the E-Business Control Center installs into the following directory (see Figure 1-1):
`bea\E-BusinessControlCenter`

In addition, when you update your WebLogic Server license to support Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, the `UpdateLicense` modifies the `bea/license.bea` file. Do not modify or move the `license.bea` file.

Figure 1-1 Top-Level Directory



Viewing and Modifying Properties in the WebLogic Server Administration Console

Because Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server runs as an instance of WebLogic Server with extended functionality, modifying WebLogic Server properties from the Administration Console changes the behavior of Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.

If you create your own Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server domain for administering your web applications, you must include the `console.war` file that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provides. The `console.war` must be located at the following pathname: `config/`*`your-domain`*`/applications/console.war`

This section describes the following tasks:

- Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain
- Determining Which Modifications Require You to Restart the Server
- Finding the Server Properties
- Changing the Listen Ports
- Changing the Message Output

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*

Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain

A WebLogic Server domain is the administrative unit that you use to manage the web applications and enterprise applications that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provides. Because you can administer a domain only from an Administration Console that is located in the domain directory tree, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provides a WebLogic Server administration console for its domain, wlcsDomain. To start the Administration Console, do the following:

1. Start Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. For more information, refer to "Starting the Server and wlcsDomain on UNIX" on page 4-2 or "Starting the Server and wlcsDomain on Windows" on page 4-3.

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

   `http:// { localhost | IP-address | hostname}:listen-port/console`

   where *`listen-port`* is the `ListenThread` number that the server reports when it successfully starts.

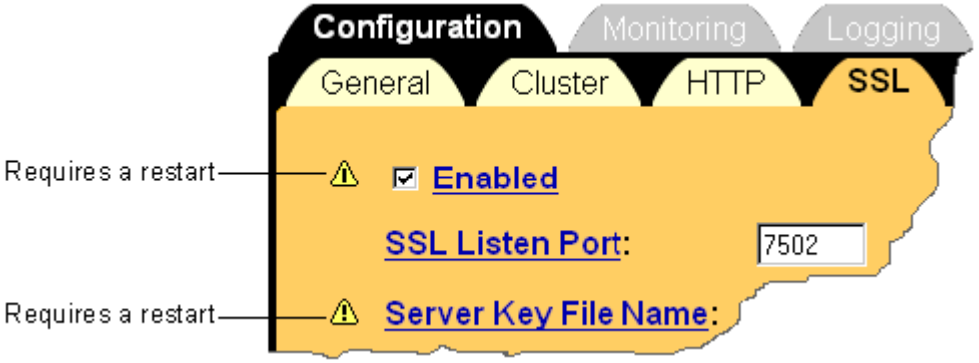
   For example, `http://127.0.0.1:7501/console` or `http://localhost:7501/console`

If you create your own Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server domain, start the server for your own domain. Then you can use the URL that this section describes to access the Administration Console that you included in the domain.

Determining Which Modifications Require You to Restart the Server

The Administration Console places a caution icon (a yellow triangle with an exclamation point inside) to indicate which properties require you to restart the server to deploy any modifications to them. (See Figure 1-2.)

Figure 1-2 The Caution Icon Indicates a Restart Is Required



Finding the Server Properties

The Administration Console uses the concept of a *server* to maintain multiple sets of properties, one of which you can apply to any given WebLogic Server instance.

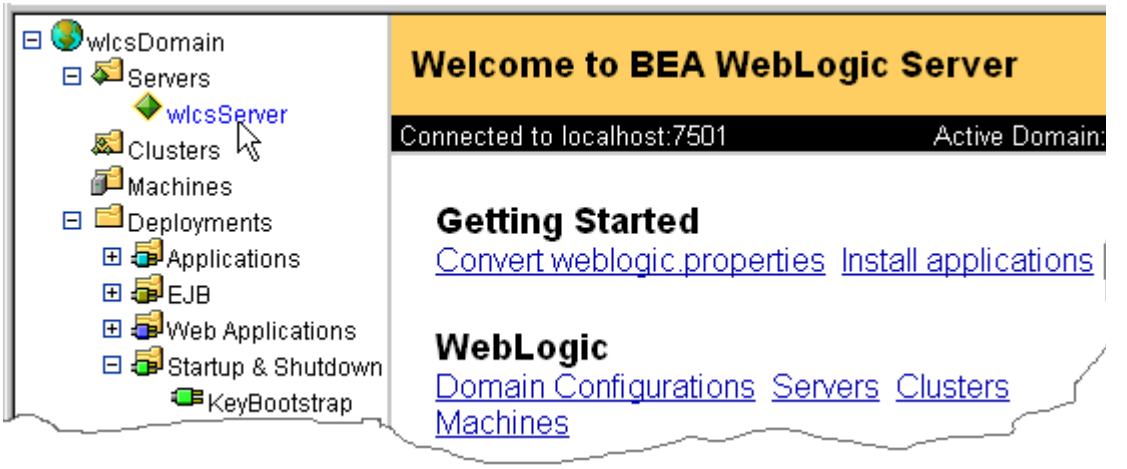
wlcsDomain defines a set of server properties named `wlcsServer`. To see the properties defined for wlcsServer, start the Administration Console (as described in Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain) and do **one** of the following:

- From the left navigation pane, click Servers → wlcsServer (see Figure 1-3).

or

- From the Home page, under WebLogic, click Servers. On the Servers page, click the wlcsServer name.

Figure 1-3 The wlcsServer Instance



For more information about starting Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, refer to Chapter 4, "Starting and Shutting Down the Server."

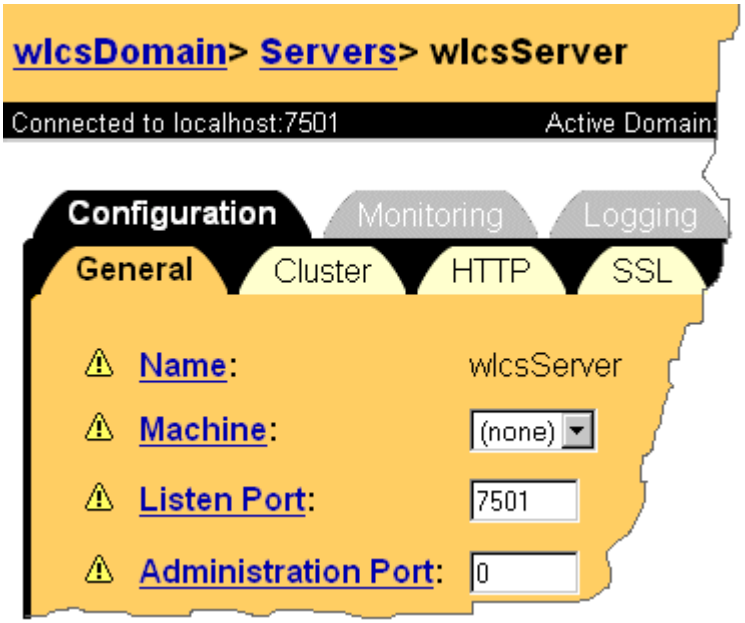
Changing the Listen Ports

You can use the Administration Console to change the ports on which wlcsServer listens for HTTP and HTTPS requests. To change the listen ports, do the following:

1. Find the server properties page as described in Finding the Server Properties.
2. On the server properties page, click the Configuration tab.

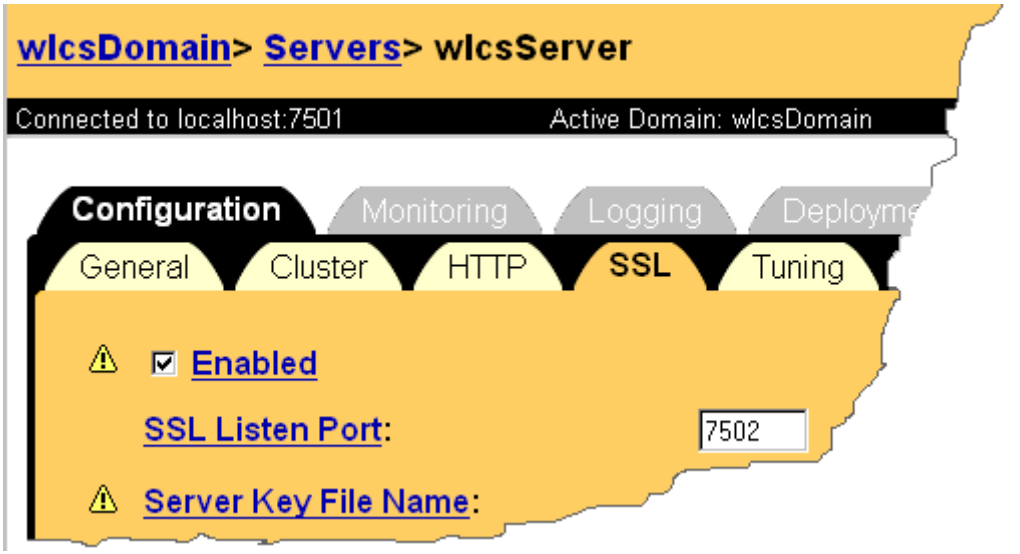
3. On the General subtab, change the value for the HTTP Listen Port (see Figure 1-4).

Figure 1-4 The HTTP Listen Port



4. On the SSL subtab, change the value for the SSL Listen Port (see Figure 1-5).

Figure 1-5 The SSL Listen Port



5. To deploy these modifications, you must restart the server. For information on shutting down and restarting the server, refer to Chapter 4, "Starting and Shutting Down the Server." We recommend that you do not use the WebLogic Server Administration Console to shut down Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.

Changing the Message Output

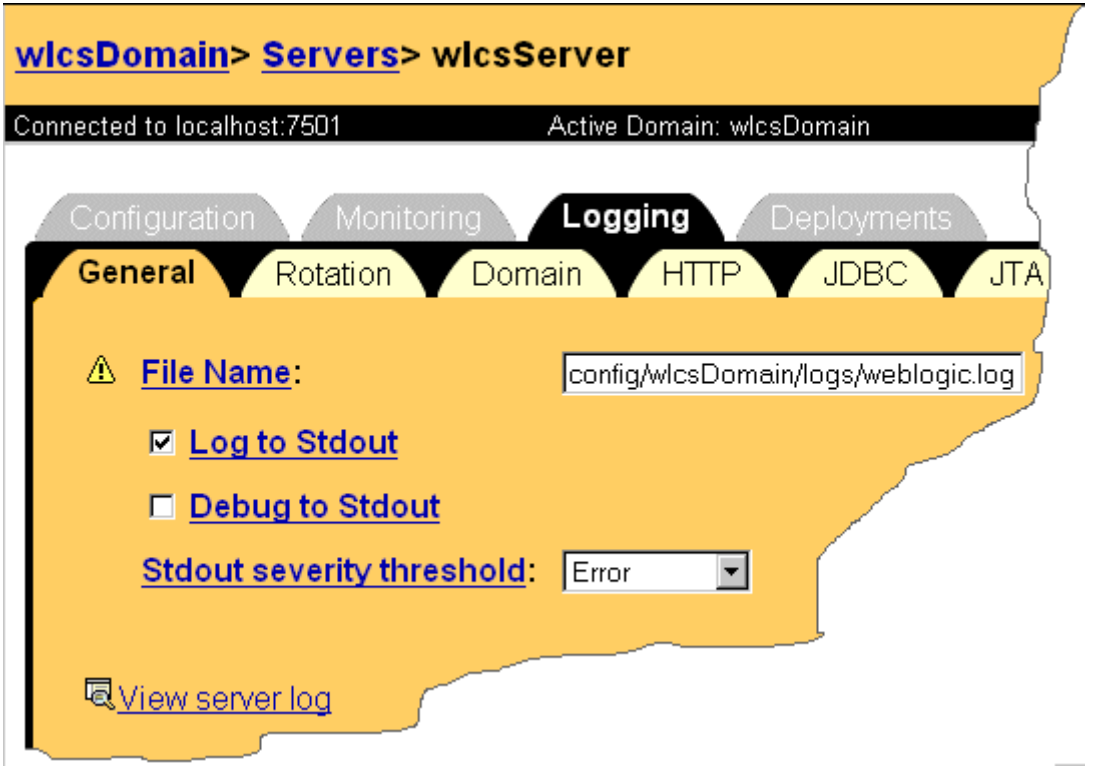
You can use the Administration Console to change the level of messages that display in the shell that runs the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server process.

To change the message output, do the following:

1. Find the server properties page as described in Finding the Server Properties.
2. On the server properties page, click the Logging tab.

3. On the General subtab, select a value from the Stdout severity threshold list (see Figure 1-6). For more information about message and log options, refer to the *Administration Console Online Help*.

Figure 1-6 Stdout Severity Threshold List



Viewing and Modifying Properties in weblogiccommerce.properties

The `WL_COMMERCE_HOME/weblogiccommerce.properties` file sets values for features specific to Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. Before you deploy the server and your enterprise applications, we recommend that you review the file.

Note: Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server refers to this file only when you start the server; **any modifications to `weblogiccommerce.properties` require that you restart the server**.

The `weblogiccommerce.properties` is divided into the following sections:

- JDBC Pool and JNDI Helpers
- CLOB Retrieval and Setting
- Personalization Attribute Loader
- Location of GIF Files
- EJB Mappings
- Campaign Service Properties
- Default Values for Portals
- User Management
- Name and Location of Documentation
- Database Timeout
- Logger Class
- Component Properties
- Payment Service
- Credit Card Security

- TAXWARE
- Debug Mode for PipelineSession
- Webflow and Pipeline Hot Deploy
- Property Cache Settings
- Catalog Cache Settings
- Cache Settings for the Discount Service
- Event Service and Behavior Tracking Parameters
- Mail Service Properties
- AdTarget Properties

JDBC Pool and JNDI Helpers

The properties in Listing 1-1 enable Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server to use JDBC pools to connect to the database. Do not change these properties.

Listing 1-1 also shows properties that define the JNDI context that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server uses to locate EJBs.

We recommend that you do not place JSPs and EJBs on separate servers. However, if you do, you must set the `commerce.configuration.splitEjbJspServer` property to `true` and enable the default context properties by removing the comment tag (#).

Listing 1-1 JDBC Pool and JNDI Helpers

```
# JDBC Pool entries

commerce.jdbc.pool.name=commercePool
commerce.jts.pool.name=commercePool
PersistenceURL=jdbc:weblogic:jts:commercePool
commerce.jdbc.pool.url=weblogic.jdbc.jts.commercePool

# JNDI helper default context setup properties. Typically you
wouldn't change these.
```

```
            # Only change these if your JNDI tree is located on another server,
            and it is not in a cluster.

            #default.context.initial.context.factory=weblogic.jndi.WLInitialC
            ontextFactory

            #default.context.provider.url=t3://localhost:7501

            #default.context.security.authentication=simple

            #default.context.security.principal=admin

            #default.context.security.credentials=adminpassword

            #default.context.security.protocol=ssl

            #default.context.authoritative=

            #default.context.batchsize=

            #default.context.dns.url=dns://somehost/wiz.com

            #default.context.language=

            #default.context.object.factories=

            #default.context.referral=

            #default.context.url.pkg.prefixes=

##################################################
# Server configuration
##################################################

# Set to true if the EJB server is not on the same cluster node as the
# JSP server.

commerce.configuration.splitEjbJspServer=false
```

CLOB Retrieval and Setting

The properties in Listing 1-2 configures the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server data repository to retrieve and set CLOBs. Set these properties to `true` only if the database JDBC driver correctly supports the `java.sql.Clob` object and methods.

Note: For Oracle and Cloudscape databases, set these properties to `false` or leave the default # (pound sign) at the beginning of the line to deactivate them. While some of these database JDBC drivers support the `java.sql.Clob` object and methods, the WebLogic Server 6.0SP1 Oracle jDriver currently has a bug and should retrieve CLOB columns through `String` methods.

Listing 1-2 CLOB Retrieval and Setting

```
# Uncomment this and set to true to use java.sql.Clob
# objects for CLOB database column retrieval. Some
# databases don't support using ResultSet.getString()
# for CLOB retrieval, which is the default behavior.

#commerce.jdbc.read.shouldUseClobs=false

# Uncomment this and set to true to use
# java.sql.ResultSet.setCharacterStream()
# to set CLOB database columns. Some databases
# don't support using ResultSet.setString()
# for CLOB setting, which is the default behavior.

#commerce.jdbc.write.shouldUseClobs=false
```

Personalization Attribute Loader

The property in Listing 1-3 retrieves additional values from the HTTP request or session object and saves them in a named location. Do not change this property.

Listing 1-3 Attribute Loader

```
# P13N Request AttributeLoaders
#p13n.request.loaders=
# P13N Session AttributeLoaders

p13n.session.loaders=com.bea.commerce.ebusiness.campaign.Shopping
CartAttributeLoader
```

Location of GIF Files

The WebLogic Commerce Server JSP templates uses the Webflow helper to generate references to images. By default, the Webflow helper references are relative to the root of the current web application.

If you locate your images outside the web application directory tree, activate the property in Listing 1-4 and provide a base URL for image pathnames.

Listing 1-4 Location of GIF Files

```
# Location of where gifs are stored.
# If not set then they are taken relative to the web app.
# commerce.gif.url.base=http://<hostname>:<port>/<path>
```

EJB Mappings

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server uses the properties in Listing 1-5 to locate EJB homes. Do not modify these properties.

Listing 1-5 EJB Mappings

```
# Entries for mapping schema group names to the appropriate home name

commerce.schemaGroupHomeName.USER=com.beasys.commerce.foundation.property.Schem
a

commerce.schemaGroupHomeName.REQUEST=com.beasys.commerce.foundation.property.Sc
hema

commerce.schemaGroupHomeName.SESSION=com.beasys.commerce.foundation.property.Sc
hema

commerce.schemaGroupHomeName.CONTENT=com.beasys.commerce.axiom.document.Documen
tSchema

commerce.schemaGroupHomeName.APPLICATION_INIT=com.beasys.commerce.foundation.pr
operty.Schema

commerce.schemaGroupHomeName.CATALOG=com.beasys.commerce.foundation.property.Sc
hema

# EJB Home Name Associations

commerce.home.property.SchemaManagerHome=\

    com.beasys.commerce.foundation.property.SchemaManager

 commerce.home.AdvisorHome=\

     com.beasys.commerce.platform.advisor.EjbAdvisorHome
```

Campaign Service Properties

The properties in Listing 1-6 configure the Campaign Service.

The first property, `campaign.rules.session.attrNames`, enables the Campaign Service to retrieve the values from the request and session objects that the Personalization attribute loader property saves. For more information about the Personalization attribute loader property, refer to "Personalization Attribute Loader" on page 1-17.

The second property, `campaign.goals.checkTime`, establishes a cache that campaigns use to determine whether they have reached their goals.

When you create a campaign, you can specify that some actions, such as displaying an ad in a placeholder, are to occur a specific number of times. Each time the action occurs, the Campaign Service updates a field in the database.

Before a Campaign Manager for WebLogic service engages an action that is associated with a campaign, for example before an ad placeholder displays an ad that is associated with a campaign, it checks with the Campaign Service to determine the following:

- Whether today's date falls within the start and stop date for the campaign.
- Whether the campaign has fulfilled its goals.

If the date is in the campaign's date range, and if the campaign has not fulfilled its goals, then the service engages the action.

You use the `campaign.goals.checkTime` to determine how frequently the Campaign Service accesses the database to determine whether it has fulfilled its goal. Each time it accesses the database, it saves the results to a cache. For example, the default setting determines that every 5 minutes (300, 000 milliseconds), the Campaign Service checks with the database to determine which goals have been fulfilled. Each time it checks with the database, it saves the results in a cache. If an ad placeholder refers to the Campaign Service to determine whether a specific ad has already reached its limit, the Campaign Service refers to the data in its cache.

A campaign can exceed its goal if the cache contains stale data.

If you want to disable the cache, provide a value of `0`.

Listing 1-6 Campaign Service Properties

```
##############################################
# Campaign Server
##############################################

# Campaign Server -- additional attributes to pull from the request and
# session and to pass to the rules engine (comma-separated, no spaces)
#campaign.rules.request.attrNames=

campaign.rules.session.attrNames=wlcs_shoppingCart

# How often should we check campaign goals, in milliseconds.

campaign.goals.checkTime=300000
```

Default Values for Portals

The properties in Listing 1-7 establish defaults for new portals. You can adjust these values based on the nature of the portal being built. You can override these defaults from the portal administration tools.

For example, the following property tells the portal component which property set to use as the default for portal users:

```
commerce.default.portal.schemaName=_DEFAULT_PORTAL_SCHEMA
```

A developer can change this value based on the property sets that are created with the Property Set Management Administration Tool.

For example, the following portal properties are for internal use and should not be altered by a developer:

```
commerce.application.minimum.build=83914
commerce.default.category.name=Home
```

Listing 1-7 Default Values for Portals

```
# Entries taken from PortalProperties
commerce.default.portal.schemaName=_DEFAULT_PORTAL_SCHEMA
commerce.default.portal.headerURL=header.jsp
commerce.default.portal.contentURL=portalcontent.jsp
commerce.default.portal.footerURL=footer.jsp
commerce.default.portal.cols=3
commerce.default.portal.suspendedURL=suspended.jsp
commerce.default.portlet.dir=portlets/
commerce.default.portlet.image.dir=portlets/images/
commerce.default.portlet.titlebarURL=titlebar.jsp
commerce.default.banner.color=#666666
commerce.default.titlebar.bgcolor=#333399
commerce.default.show.borders=false
commerce.default.content.bgcolor=#CCCCCC
commerce.default.titlebar.font.color=#FFFFFF
commerce.default.body.bgcolor=#FFFFFF
commerce.default.profile.group.property=portal.profile.group
commerce.default.portal.password=guest
commerce.application.minimum.build=83914
commerce.default.category.name=Home
commerce.portalservicemanager.fast.portal.get=true
```

User Management

Listing 1-8 illustrates the section of `weblogiccommerce.properties` that specifies properties for user management. The first set of properties map table names. Do not modify these properties.

The set of properties under `# RDBMS Realm` determine which database type the reference implementation uses for storing user information. Modify this section only if you use the reference implementation for user management. If you use an LDAP server for user management, you do not need to modify this section. For more information, see "Using WebLogic Realms" under "Creating and Managing Users" in the *Guide to Building Personalized Applications.*

Listing 1-8 User Management

```
##############################################

# User Management

##############################################

commerce.usermgmt.UserTable=WLCS_USER
commerce.usermgmt.GroupTable=WLCS_GROUP
commerce.usermgmt.UserGroupHierarchyTable=WLCS_USER_GROUP_HIERARCHY
commerce.usermgmt.GroupHierarchyTable=WLCS_GROUP_HIERARCHY
commerce.usermgmt.EntityIdTable=WLCS_ENTITY_ID
commerce.usermgmt.ProfileTypeTable=WLCS_UNIFIED_PROFILE_TYPE



# RDBMS Realm
#------Oracle thin Driver---------#
#commerce.usermgmt.RDBMSRealm.driver=oracle.jdbc.driver.OracleDriver
#commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:oracle:thin:@@ORACLE_SERVER@:@ORACLE_P
ORT@:@ORACLE_SID@
#commerce.usermgmt.RDBMSRealm.dbUser=@ORACLE_USER@
#commerce.usermgmt.RDBMSRealm.dbPassword=@ORACLE_PASSWORD@

#------WebLogic jDriver for Oracle 8.1.5---------#
#commerce.usermgmt.RDBMSRealm.driver=weblogic.jdbc.oci.Driver
#commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:weblogic:oracle
#commerce.usermgmt.RDBMSRealm.dbServer=@ORACLE_NET_SERVICE_NAME@#commerce.userm
gmt.RDBMSRealm.dbUser=@ORACLE_USER@
#commerce.usermgmt.RDBMSRealm.dbPassword=@ORACLE_PASSWORD@

#--------------Cloudscape------------------#
commerce.usermgmt.RDBMSRealm.driver=COM.cloudscape.core.JDBCDriver
commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:cloudscape:Commerce;create=true;autocom
mit=false
commerce.usermgmt.RDBMSRealm.dbUser=none
commerce.usermgmt.RDBMSRealm.dbPassword=none
```

Name and Location of Documentation

Listing 1-9 includes the following properties that configure the WebLogic Personalization Server Content Management service:

- `commerce.home.content.*` and `commerce.home.document.*` specify JNDI home names for EJBs and provide constants for the `com.beasys.commerce.content.ContentHelper` class.

  `commerce.home.content.ContentManagerHome` provides the default value for the `contentHome` parameter in the `<cm:select>` and `<cm:selectById>` JSP tags. For information about Content Management JSP tags, refer to "JSP Tag Library Reference" in the *Guide to Building Personalized Applications.*

- `commerce.content.defaultReadOnly` controls the default read/write state of objects that Content Management JSP tags return. Do not modify this property.

- `commerce.content.cache.useSoftHashMap` controls whether the `ContentCache` object uses `SoftReferences` to maintain the cache. Modify this property only to resolve memory issues.

- `commerce.xml.entity.basePath` provides the `com.beasys.commerce.axiom.util.P13NEntityResolver` object with the location of DTDs. Do not modify this property. For more information, refer to the Javadoc for the `com.beasys.commerce.axiom.util.P13NEntityResolver` class.

Listing 1-9 Name and Location of Documentation

```
################################################
# Documentation name and location properties
################################################
commerce.p13n.document.dev.name=index.html
commerce.p13n.document.dev.version=32
commerce.p13n.document.dev.docRootDir=docs
commerce.home.content.ContentHome=com.beasys.commerce.axiom.document.Document
```

```
commerce.home.content.ContentManagerHome=com.beasys.commerce.axiom.document.Doc
umentManager

commerce.home.content.ContentSchemaHome=com.beasys.commerce.axiom.document.Docu
mentSchema

commerce.home.document.DocumentHome=com.beasys.commerce.axiom.document.Document

commerce.home.document.DocumentManagerHome=com.beasys.commerce.axiom.document.D
ocumentManager

commerce.home.document.DocumentSchemaHome=com.beasys.commerce.axiom.document.Do
cumentSchema

commerce.content.defaultReadOnly=true

commerce.content.cache.useSoftHashMap=false


commerce.xml.entity.basePath=D:/WebLogicCommerce3.5/lib/dtd
```

Database Timeout

The properties in Listing 1-10 specify the amount of time that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server waits to obtain a connection to the database pool before timing out.

Listing 1-10 Database Timeout

```
##############################################

# Time out properties for weblogic database connection

##############################################

connection.TimeOutPeriod=999999999

waitForConnection=true
```

Logger Class

The properties in Listing 1-11 do the following:

- Specify that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server use the WebLogic Server facility for logging messages instead of sending messages to the shell that runs the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server process.
- Load the class that sends messages to the WebLogic Server logging facility.

Listing 1-11 Properties for Logging Messages

```
##############################################

# Logger class

##############################################

commerce.log.class=com.bea.commerce.platform.logging.NonCatalogLo
g

commerce.log.display.deprecated=true
```

Component Properties

The properties in Listing 1-12 are used by the Bean-Managed Persistence (BMP) layer for data persistence. Do not modify these properties.

Listing 1-12 Component Properties

```
##############################################

# Properties required for components

##############################################


# Use the BMP plugin by default

DefaultPersistenceHelperPlugin=com.beasys.commerce.foundation.plugin.bmp.BMPPer
sistenceHelperPlugin
```

Payment Service

The properties in Listing 1-13 configure the payment service, including the CyberCash credit card validation service.

For more information about this section, refer to "Configuration Activities for Using CyberCash" under "Payment Services" in the *Guide to Managing Purchases and Processing Orders*.

Listing 1-13 Payment Service

```
####################################################

# Properties required for the payment component

####################################################

#
```

```
# This property defers payment authorization to the administration tools.

# If set to true, all payment service authorization calls are disabled

# and payment transactions are persisted in a RETRY state. Payments must

# then be reauthorized through the payment administration tool.

#

commerce.payment.defer.authorization=true


#

# CyberCash configuration files contain CyberCash-specific data, such as a

# merchant-id and merchant hash secret. The specific properties in the
configuration

# files depend upon the payment model assigned to a merchant by his/her financial

# institution. The two files declared below are example files and are provided

# for demonstration purposes ONLY. MERCHANTS MUST AQUIRE A CYBERCASH CONFIGURATION

# FILE FROM CYBERCASH. These will be furnished by CyberCash as part of the
# merchant agreement. Once a merchant has a CyberCash configuration file, the
# property below must be replaced with the location of the configuration file.

#

# Example:
CyberCashConfigFile=c:/merchant/config/file/location/merchant_conf-terminal

# This file may be used for testing terminal based payment models.

CyberCashConfigFile=D:/WebLogicCommerceServer3.2/eval/common/CyberCash/conf/mer
chant_conf-terminal

# This file may be used for testing host based payment models.

#CyberCashConfigFile=D:/WebLogicCommerceServer3.2/eval/common/CyberCash/conf/me
rchant_conf-host

#

# Properties below represent the different payment models provided by CyberCash.

#

# Terminal based models.
```

```
PaymentModel=AUTO_MARK_AUTO_SETTLE
#PaymentModel=AUTO_MARK_AUTO_SETTLE_AVS
#PaymentModel=AUTO_MARK_MANUAL_SETTLE
#PaymentModel=AUTO_MARK_MANUAL_SETTLE_AVS
#PaymentModel=MANUAL_MARK_AUTO_SETTLE
#PaymentModel=MANUAL_MARK_AUTO_SETTLE_AVS
#PaymentModel=MANUAL_MARK_MANUAL_SETTLE
#PaymentModel=MANUAL_MARK_MANUAL_SETTLE_AVS
# Host based models.
#PaymentModel=HOST_AUTH_CAPTURE
#PaymentModel=HOST_AUTH_CAPTURE_AVS
#PaymentModel=HOST_POST_AUTH_CAPTURE
#PaymentModel=HOST_POST_AUTH_CAPTURE_AVS
```

Credit Card Security

The properties in Listing 1-14 configures security services for credit card transactions. For information about this section, refer to "Credit Card Security Service" under "Payment Services" in the *Guide to Managing Purchases and Processing Orders.*

Listing 1-14 Credit Card Security

```
#####################################################
# Properties required for Security Services
#####################################################
#
# Credit card encryption services are turned on by setting this property to true.
# Commenting out this property or setting it to false will disable credit card
```

```
# encryption.
#
is.encryption.enabled=true

#
# The name of the security table and column names for public and private encryption
# keys can be specified using the properties below.
#
security.table.name=WLCS_SECURITY
security.backup.table=WLCS_SECURITY_BACKUP
public.key.column.name=PUBLIC_KEY
private.key.column.name=PRIVATE_KEY

#
# The key bit size desired.  Key bit length and length of data that can be encrypted
# are related as follows:
#    KEY BIT LENGTH(bits)           DATA LENGTH (bytes)
#        512                             53
#       1024                            117
#       2048 (MAX LENGTH)               245
key.bit.size=1024

#
# This optional parameter specifies the private key password used to decrypt the
# encrypted credit card encryption private key.  WARNING: Setting this property
# will start up the server without prompting for a password.
private.key.password=WLCS
```

TAXWARE

The properties in Listing 1-15 configure the TAXWARE tax calculation service. For information about this section, refer to "TAXWARE Configuration and Deployment" under "Taxation Services" in the *Guide to Managing Purchases and Processing Orders*.

Listing 1-15 TAXWARE

```
######################################################################
# TAXWARE TAX SERVICE  PROPERTIES
# ------------------------------------------------
# For instructions on individual properties in this
# section, Please refer to the Taxware SalesUse and Verazip manuals.
#
######################################################################


######################################################################
# JNDI name of the TaxCalculator Session Bean
# ------------------------------------------------------------------

tax.calculator.jndi.name =
com.beasys.commerce.ebusiness.tax.taxware.TaxwareTaxCalculator


######################################################################
# Currency for Tax Calculation (Taxware only supports USD)
# ------------------------------------------------------------------

tax.currency = USD


######################################################################
```

```
# ShipFrom Address
# ----------------------------------------------------------------
# ShipFrom Address is address from where goods are shipped
# Please review Taxware documentation when setting these properties
#
shipfrom.countycode=000
shipfrom.state=MA
shipfrom.city=SALEM
shipfrom.zip=01970
shipfrom.geocode=00
shipfrom.country=USA

#####################################################################
# Order Acceptance Address
#-----------------------------------------------------------------
# OrderAcceptance is the address where orders are accepted
# Please review Taxware documentation when setting these properties
#
orderacceptance.countycode=000
orderacceptance.state=MA
orderacceptance.city=SALEM
orderacceptance.zip=01970
orderacceptance.geocode=00
orderacceptance.country=USA

#####################################################################
# Order Origin Address
```

```
#-----------------------------------------------------------------
# Order Origin is the address where orders are Originated
# Please review Taxware documentation when setting these properties
#
orderorigin.countycode=000
orderorigin.state=MA
orderorigin.city=SALEM
orderorigin.zip=01970
orderorigin.geocode=00
orderorigin.country=USA

#####################################################################
# Point of title passage
# ----------------------------------------------------------------
# Location at which legal title has transferred to purchaser

#titlepassage=shipto
titlepassage=shipfrom

#####################################################################
# Company Identification
#-----------------------------------------------------------------
# User Defined company identification to access information
# for tax calculating and reporting

companyId=companyId
```

```
#####################################################################
# TaxType
#-------------------------------------------------------------------
# Type of tax to be calculated

#taxtype=use
#taxtype=rental
#taxtype=consumeruse
#taxtype=services
taxtype=sales

#####################################################################
# TaxSelParm
#-------------------------------------------------------------------
# Taxselparm to decide jurisdiction while calculating
# if value is 2 Calculate tax only
# if value is 3 Determine jurisdiction and calculate taxes
#taxselparm=2
taxselparm=3
```

Debug Mode for PipelineSession

The properties in Listing 1-16 establish a debug mode, in which you can determine whether you have assigned the correct scope to an attribute. To see the debug messages, you must set the message log to display debug messages.

For more information about attribute scopes, refer to "Attribute Scoping" under "Extending Webflow and Pipelines" in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Listing 1-16 Debug Mode for PipelineSession

```
#################################################################
# Enable Debug Mode for PipelineSession
#---------------------------------------------------------------
# This checks if the attribute that the user is trying to set to the
# PipelineSession is serializable or not
# If the attribute is not serializable then a message and exception
# stack trace is displayed

pipelineSession.debug=false
```

Webflow and Pipeline Hot Deploy

The properties in Listing 1-17 determine whether you can change your web application's Webflow and Pipeline without restarting Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.

For more information about changing Webflow and Pipeline, see "Customizing Webflow and Pipelines" in the *Guide to Managing Presentation and Business Logic*.

Listing 1-17 Webflow and Pipeline Hot Deploy

```
##########################################
# Enable webflow and pipeline hot deploy
#----------------------------------------
webflow.hotdeploy.enable=false
pipeline.hotdeploy.enable=false
```

Property Cache Settings

The properties in Listing 1-18 configure the property cache, which decreases the amount of time needed to access user, group, and other properties. For information on the property cache, refer to "Enable Property Caching" in the *Performance Tuning Guide*.

Listing 1-18 Property Cache Settings

```
##############################################

# Cache settings

# cache of unified profile types, 1 hour, 100 entries
unifiedProfileTypeCache.ttl=3600000
unifiedProfileTypeCache.capacity=100
unifiedProfileTypeCache.enabled=true

# cache of entity properties, 10 minutes, 500 entries
entityPropertyCache.ttl=600000
entityPropertyCache.capacity=500
entityPropertyCache.enabled=true

# cache of property default values, 1 hour, 100 entries
propertyDefaultCache.ttl=3600000
propertyDefaultCache.capcity=100
propertyDefaultCache.enabled=true

# cache of entity ids, 1 hour, 500 entries
entityIdCache.ttl=3600000
entityIdCache.capacity=500
entityIdCache.enabled=true

# cache of explicit properties, 10 minutes, 100 entries
directPropertyManager.ttl=600000
directPropertyManager.capacity=100
directPropertyManager.enabled=true

# cache of ConfigurableEntity methods, 1 hour, 100 entries
ConfigurableEntityMethodCache.ttl=3600000
ConfigurableEntityMethodCache.capacity=100
ConfigurableEntityMethodCache.enabled=true

# cache of ldap usernames, 1 hour, 100 entries
ldapUserCache.ttl=3600000
```

```
ldapUserCache.capacity=100
ldapUserCache.enabled=true

# cache of ldap group names, 1 hour, 10 entries
# (raise this if you have a lot of groups in ldap)
ldapGroupCache.ttl=3600000
ldapGroupCache.capacity=10
ldapGroupCache.enabled=true

#cache of session values
_sessionCache.ttl=900000
_sessionCache.capacity=10000
_sessionCache.enabled=true

#cache of values across multiple users
_globalCache.ttl=600000
_globalCache.capacity=1000
_globalCache.enabled=true
```

Catalog Cache Settings

The properties in Listing 1-19 define cache settings for the product catalog. For more information about this section, refer to "Improving Catalog Performance by Optimizing the Catalog Cache" under "Catalog Administration Tasks" in the *Guide to Building a Product Catalog*.

Listing 1-19 Catalog Cache

```
# Cache entries
ProductItemCache.ttl=21600000
ProductItemCache.capacity=10000
ProductItemCache.enabled=true

CategoryCache.ttl=86400000
CategoryCache.capacity=1000
CategoryCache.enabled=true
```

Cache Settings for the Discount Service

The properties in Listing 1-20 configure the cache for the Discount Service, which decreases the amount of time the Order and Shopping Cart services need to calculate order and price information that include discounts. For information on the discount-service cache, refer to "Adjust Caching for the Discount Service" in the *Performance Tuning Guide*.

Listing 1-20 Discount Association Service Cache Settings

```
# Discount Service cache entries
# cache of all discounts, 5 minutes, 100 entries

discountCache.ttl=300000
discountCache.capacity=100
discountCache.enabled=true

# the global discount cache, 5 minutes, 10 entries

globalDiscountCache.ttl=300000
globalDiscountCache.capacity=10
globalDiscountCache.enabled=true
```

Event Service and Behavior Tracking Parameters

The properties in Listing 1-21 set parameters for the Event and Behavior Tracking services. For example, they register custom event listeners and determine whether the listener is synchronous or asynchronous.

The properties also register both synchronous and asynchronous listeners for campaigns. If you do not use Campaign Manager for WebLogic, you can remove or comment-out the following listeners:

- `com.bea.commerce.campaign.internal.CampaignEventListener`
- `com.bea.commerce.campaign.internal.AsyncCampaignEventListener`

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

Listing 1-21 Event Service Parameters

```
##############################################
# Event Service
##############################################

#

# Listeners for EventService's handler.  Each listener expresses interest
# in what Event Types it can handle.  Events of that type sent to the
# EventService will be dispatched to the handler.
# This property should be a comma-delimited list of (fully qualified)
# class names. Classes listed here must implement the interface
# com.bea.commerce.platform.events.EventListener
# and must also have a no-args (default) constructor.
#
# To enable behavior tracking, add
# com.bea.commerce.platform.tracking.listeners.BehaviorTrackingListener
# to this list.
#
# To enable asynchronous event handling, add
# com.bea.commerce.platform.events.listeners.AsynchronousListener
# to this list.

eventService.listeners=\

    com.bea.commerce.campaign.internal.CampaignEventListener,\

    com.bea.commerce.platform.events.listeners.AsynchronousListener

#    com.bea.commerce.platform.tracking.listeners.BehaviorTrackingListener
#    com.bea.commerce.platform.events.listeners.DebugEventListener

##############################################
# Behavior Tracking
##############################################
#
# Events to be persisted to the behavior tracking database.  These
# are used when uncommenting the BehaviorTrackingListener class in
# the eventService.listeners property above.

#
# This is the list of event types that will be persisted to the
# behavior tracking database tables.  Events in this list must
```

```
# extend the com.bea.commerce.platform.tracking.events.TrackingEvent
# class.
#

behaviorTracking.persistToDatabase=\

    AddToCartEvent,\

    BuyEvent,\

    CampaignUserActivityEvent,\

    ClickContentEvent,\

    ClickProductEvent,\

    ClickCampaignEvent,\

    DisplayContentEvent,\

    DisplayProductEvent,\

    DisplayCampaignEvent,\

    PurchaseCartEvent,\

    RemoveFromCartEvent,\

    RuleEvent,\

    SessionBeginEvent,\

    SessionEndEvent,\

    SessionLoginEvent,\

    UserRegistrationEvent

# For persistence of tracking events to the database, a cache is used
# to accumulate events (rather than generating a single database hit
# for each event).  The events are swept to the database based
# on size and time.  Every checkIntervalSec seconds, a size check
# is performed, and if the cache size is greater than the maxCount
# threshold, the events are swept to the database.  Every maxAgeSec
# seconds, a sweep of the events in the cache is forced.
#
# The number of events are cached before a size-based sweep of
# tracking events to the database can be triggered.
#

behaviorTracking.cache.maxCount=200
```

```
# Every checkIntervalSec seconds, the cumulative cache size
# will be checked.  If this check yields a size greater than
# behaviorTracking.cache.maxCount, the events will be flushed
# to the database.
#

behaviorTracking.cache.checkIntervalSec=5

# The time to wait before forcing a flush to the database
# in seconds.  This is the longest amount of time that an event
# will exist in any cache.
#

behaviorTracking.cache.maxAgeSec=30

# Database connection pool name to use for behavior tracking data

behaviorTracking.database.connectionPool=commercePool

##############################################
# Asynchronous Event Service
##############################################

#
# Listeners for the asynchronous side of the event service.  Each listener
# expresses interest in what Event Types it can handle.  Events of that
# type sent to the EventService will be dispatched to the handler.
# This property should be a comma-delimited list of (fully qualified)
# class names. Classes listed here must implement the interface
# com.bea.commerce.platform.events.EventListener
# and must also have a no-args (default) constructor.
#

asynchronousHandler.listeners=\

    com.bea.commerce.campaign.internal.AsyncCampaignEventListener
```

Mail Service Properties

The first property in Listing 1-22 determines name of the SMTP host that the Mail Service sends to the JavaMail API.

The last two properties redirect requests to a proxy server. Use these properties in clustered environments.

The remaining properties determine how campaigns address e-mails that they send as part of a scenario action. For more information, refer to "Setting Properties for the Mail Service," under "Setting Up and Sending Email for Campaigns" in the *Guide to Developing Campaign Infrastructure*.

Listing 1-22 Mail Service Properties

```
################################################
# MailService and MailAction properties

# SMTP Host for the MailService to use
mail.smtpHost=walawala.sprockets.com

# A default from-address, if none is provided to the MailAction
mailAction.defaultFromAddress=acme@sprockets.com

# The property set to query for the user's email address
mailAction.emailPropertySet=Customer Properties

# The name of the property to query for the user's email address
mailAction.emailPropertyName=email

# The property set that contains the email opt-in property
mailAction.optInPropertySet=Demographics

# The name of the email opt-in property. If a user sets this to
# false, MailAction will not send any emails to that user.
mailAction.optInPropertyName=Email_Opt_In

# The name of the server that contains the email-generating JSP's
# (overriding this should only be necessary in a cluster)
#mailAction.jspHost=localhost

# The port of the server that contains the email-generating JSP's
# (overriding this should only be necessary in a cluster)
#mailAction.jspPort=7501
```

AdTarget Properties

The properties in Listing 1-23 support ad placeholders and the `<ad:adTarget>` JSP tag.

The `adtargettag.rendering` property specifies the class that generates the HTML elements that the browser requires to display an ad through the `<ad:adTarget>` JSP tag. WebLogic Personalization Server provides only one class. If you write your own class, it must implement the `bea.commerce.platform.ad.adContentProvider` interface. For more information, refer to the WebLogic Personalization Server Javadoc.

The `adtargettag.eventtracking` property determines the class that the adTarget tag uses to access the Event Service. The WebLogic Personalization Server installer configures this property.

The `adtargettag.rendering.*` property specifies classes to generate HTML for additional MIME types. For more information, refer to "Supporting Additional MIME Types" under "Working with Ad Placeholders" in the *Guide to Building Personalized Applications.*

The `adservicebean.display.flush.size` property determines the number of display counts that are stored in memory before updating the database. The Campaign Service uses display counts to determine whether a campaign has met its end goals. Each time an ad placeholder displays an ad as a result of a scenario action, the Campaign Service updates the display count. With the default setting of 10, the Campaign Service does not update the display count in the database until 10 ads have displayed as a result of one or more scenario actions placing queries in ad placeholders.

If the server crashes before the Campaign Server flushes this display-count buffer to the database, you can lose display-count updates, up to the value of the `adservicebean.display.flush.size` property.

For sites with high traffic, increase this number to a range of 50 to 100.

Listing 1-23 AdTarget Properties

```
##################################################

# AdTargetTag Properties
adtargettag.rendering=com.bea.commerce.platform.ad.AdClickThruServlet
```

```
# This is the class that implements the AdEventTracker interface
# and is used to raise events

adtargettag.eventtracking=com.bea.commerce.campaign.AdTracking



# Additional classes to render content based upon mime type
# To use replace the "text.html" with the mime type, replacing any
#   '/' characters with '.'
# Place the name of the java class that handles the mime type after the '='

#adtargettag.rendering.text.html=

# This value determines how many display update counts are cached before
# we flush them to the database

adservicebean.display.flush.size=10
```

CHAPTER

2 The Reference Domain

A WebLogic Server domain is the administrative unit that you use to manage one or more web applications and/or enterprise applications. Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server includes a sample (reference) domain, which includes an enterprise application that your developers can use as a starting point for developing your own enterprise application. The following sections in this topic describe the reference domain and its contents:

- About the Reference Domain
- About the Reference Enterprise Application
- About the Example Portal Deployment Descriptor
- About the e-Commerce Deployment Descriptor
- The weblogic.xml File

Before you continue, we recommend that you read the following:

- "Web and Enterprise Applications for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server" in the *Product Family Overview*.
- The "Application Assembly and Deployment" chapter of the *Java™ 2 Platform Enterprise Edition Specification, v1.3*.

About the Reference Domain

Because WebLogic Server specifies that all domains must reside in a directory named `config`, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server installs its reference domain in `WL_COMMERCE_HOME/config/wlcsDomain`.

A `config` directory can contain any number of domains. You can place a `config` directory in any location that can access (and is accessible to) WebLogic Server and Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server services.

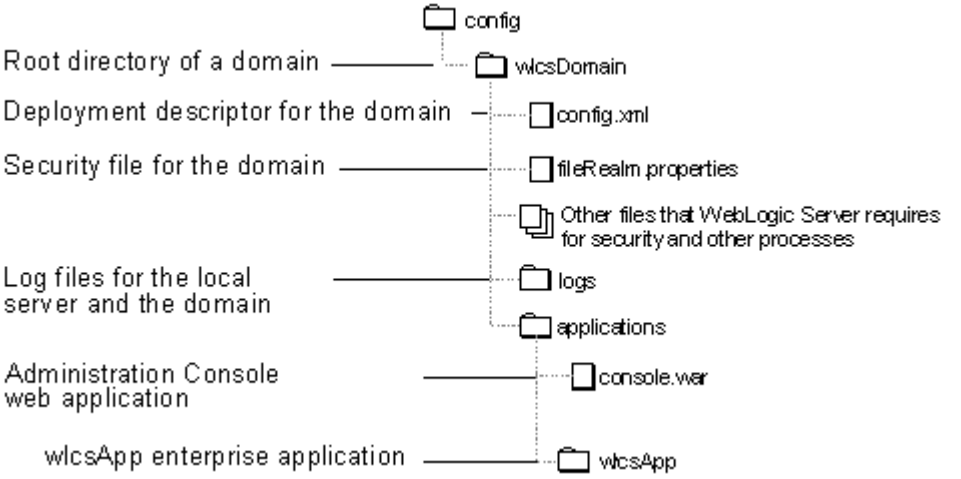
The domain that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server installs, `wlcsDomain`, contains the following (see Figure 2-1):

- `config.xml`, the configuration file for the domain. Among other things, `config.xml` notifies WebLogic Server of which EJBs, web applications, and enterprise applications 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.
- `fileRealm.properties`, which describes ACLs and an encrypted version of the password needed to start the Administration Server for the current domain. Even if you use some other security realm to authenticate users, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server uses `fileRealm.properties` to describe the administrator ACL and password.
- `logs`, which contains the local log files for the server that is installed on the current machine and domain-level log files. For more information, refer to "Using Log Messages to Manage WebLogic Servers" in the *BEA WebLogic Server Administration Guide*.
- `applications`, which contains `console.war`, the WebLogic Administration Console for the domain. You must deploy this `console.war` (or a copy) in any domain that you develop under the following pathname: `config/`*`your-domain`*`/applications/console.war`

  The `applications` directory also contains `wlcsApp`, the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server

enterprise application. For more information, refer to “About the Reference Enterprise Application” on page 2-3.

Figure 2-1 The config and wlcsDomain Directories



About the Reference Enterprise Application

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provides a reference enterprise application, wlcsApp, as a starting point from which you develop your own enterprise application. The wlcsApp directory tree contains the following (see Figure 2-2):

- `META-INF`, which contains `application.xml`, the deployment descriptor for the enterprise application. For more information, refer to “The wlcsApp Deployment Descriptor” on page 2-7.
- EJB JAR files used by the enterprise application. You must deploy all of the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server EJBs with any web applications that you develop. For more information, refer to “Deploying Your EJBs or Web Applications” on page 3-7.

- `tools`, which contains the Administration Tool web application that you use to administer such Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server features as portals, personalization rules, users and groups, and the product catalog. You must deploy this web application with any web applications that you develop.

Note: The Admin Tool web application and the WebLogic Server Administration Console are two separate web applications with separate functions. In addition, Admin Tool must be deployed inside an enterprise application with your `exampleportal` and `wlcs` web applications.

- `defaultWebApp`, which responds to any HTTP request that cannot be resolved to another deployed web application. For example, the URL `http://localhost:7501/` accesses the default web application. WebLogic Server requires that each domain contains a default web application. You can use this default web application to deploy resources that do not belong to any other web application. For example, you can use it to provide a home page that provides access to the other web applications.

  The `defaultWebApp` also contains JSPs and Java files that provide additional examples of implementing Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server features.

- `exampleportal`, which contains the example portal web application. We recommend that you make a backup copy of this directory or place it under source control and then use its JSPs and other files as a starting point for your own portal web application.

  The `WEB-INF` subdirectory contains the following:

  - `web.xml`, which is the web application's deployment descriptor. For more information, refer to "About the e-Commerce Deployment Descriptor" on page 2-13.
  - `*.tld`, JSP tag libraries for portals and personalization features. You must deploy these tag libraries in the `WEB-INF` directory for your web application.
  - `classes`, which contains compiled Java classes.

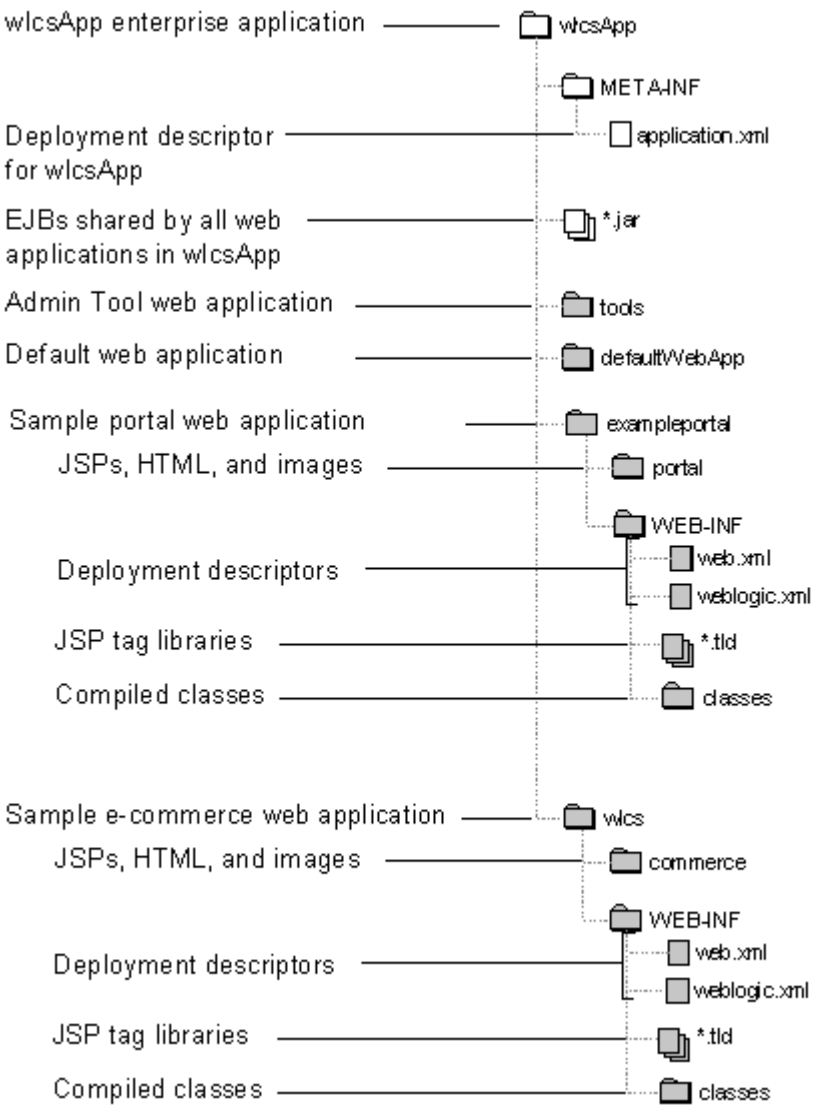
  The wlcsDomain configuration file names this as the default web application for the domain.

- `wlcs`, which contains the sample e-commerce web application. We recommend that you make a backup copy of this directory or place it under source control and then use its JSPs and other files as a starting point for your own e-commerce web application.

  The `WEB-INF` subdirectory contains the following:

  - `web.xml`, which is the web application's deployment descriptor. For more information, refer to "About the e-Commerce Deployment Descriptor" on page 2-13.
  - `weblogic.xml`, which maps global enterprise application security roles to `wlcs` security roles.
  - `*.tld`, JSP tag libraries for e-commerce features. You must deploy these tag libraries in the `WEB-INF` directory for your web application.

Figure 2-2 Directory Structure of wlcsApp Enterprise Application



The wlcsApp Deployment Descriptor

The wlcsApp deployment descriptor conforms to the J2EE specification, naming components and declaring the global security roles that apply to all components in the enterprise application.

To conform to J2EE specifications, the deployment descriptor is named `application.xml` and is located in `WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp/META-INF` where `wlcsApp` is the root of the enterprise application.

This section describes the following elements in the deployment descriptor:

- Opening Commands and Declarations
- The Root and Application Name Elements
- Module Elements
- Global Security Role Elements

For information about other elements in the deployment descriptors, refer to the *Performance Tuning Guide*.

For more information about enterprise application deployment descriptors, refer to the "Application Assembly and Deployment" chapter of the *Java™ 2 Platform Enterprise Edition Specification, v1.3*.

Opening Commands and Declarations

The deployment descriptor starts with an optional processing command:

```
<?xml version="1.0"  encoding="UTF-8"?>
```

Then it declares its DTD:

```
<!DOCTYPE application PUBLIC '-//Sun Microsystems, Inc.//DTD J2EE
Application 1.2//EN'
'http://java.sun.com/j2ee/dtds/application_1_2.dtd'>
```

The Root and Application Name Elements

The root element is named <application>, and an </application> tag closes it.

WebLogic Server uses the following element to identify the enterprise application:

```
<display-name>name</display-name>
```

You can also use the following element to document the purpose and scope of the enterprise application:

```
<description>text description</description>
```

Module Elements

Each component in an enterprise application must be declared within a module element.

The deployment descriptor uses following element syntax to name each web application:

```
<module>

  <web>

    <web-uri>URI of a web application file,
    relative to the top level of the enterprise application
    </web-uri>

    <context-root>Root directory of the web application
    </context-root>

  </web>

</module>
```

It uses the following element syntax to name each shared EJB JAR:

```
<module>

 <ejb>pathname of the JAR file that contains the EJB</ejb>

</module>
```

Global Security Role Elements

The deployment descriptor uses the following element syntax to declare security roles that are valid for all components in the enterprise application:

```
<security-role>

    <description>text displayed in the Administration Console
    </description>

    <role-name>name of role</role-name>

 </security-role>
```

About the Example Portal Deployment Descriptor

The portal (`exampleportal`) web application uses a deployment descriptor, formatted in XML, to specify the following information:

- Opening Commands and Declarations
- The Root Element and Application Description
- Precompiling JSPs
- Servlet Registration and Mapping
- MIME-Type Mapping

For more information about the XML elements that this section describes, refer to "web.xml Deployment Descriptor Elements" in *Developing WebLogic Server Applications*.

Opening Commands and Declarations

The deployment descriptor starts with an optional processing command:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
```

Then it declares its DTD:

```
<!DOCTYPE application PUBLIC '-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

The Root Element and Application Description

The root element of the deployment descriptor is named `<web-app>`, and a `</web-app>` tag closes it.

The following element documents the name of the web application, as specified in *Java™ 2 Platform Enterprise Edition Specification, v1.3*:

```
<display-name>name of web application</display-name>
```

WebLogic Server does not use this element to identify the web application. Instead, it uses the module name that you specify in the enterprise application's `application.xml` file. For more information, refer to "Module Elements" on page 2-8.

Precompiling JSPs

The following element determines whether WebLogic Personalization Server compiles all JSPs in the web application when you start the server:

```
<context-param>
     <param-name>weblogic.jsp.precompile</param-name>
     <param-value>false</param-value>
</context-param>
```

By default, the element sets the value to `false`, which deactivates the JSP precompile option. With this option deactivated, the server starts quickly but must compile each new or modified JSP when you access it, causing a significant delay the first time you request a new or modified JSP.

When you activate the precompile option, the server startup process checks for new or modified JSPs in the web application and compiles them. Activating the precompile option can cause a significant delay in server startup if you have modified or added JSPs but avoids delays when you access a new or modified JSP for the first time.

Servlet Registration and Mapping

The following element syntax registers a servlet that the web application uses:

```
<servlet>
     <servlet-name>name that WebLogic Server uses when managing
     the servlet
     </servlet-name>
    <servlet-class>name of the class file that implements
    the servlet
    </servlet-class>
</servlet>
```

The example portal web application includes several `<servlet>` elements to register several servlets.

The following element syntax defines a mapping between a servlet and a URL pattern:

```
<servlet-mapping>
   <servlet-name>name that WebLogic Server uses when managing
  the servlet</servlet-name>
   <url-pattern>pattern used to resolve URLs</url-pattern>
</servlet-mapping>
```

MIME-Type Mapping

The following `<mime-mapping>` element defines a mapping between an extension and a MIME type:

```
<mime-mapping>
    <extension>
      name of extension
    </extension>
    <mime-type>
      A string describing the defined mime type
    </mime-type>
</mime-mapping>
```

About the e-Commerce Deployment Descriptor

The e-commerce (wlcs) web application uses a deployment descriptor, formatted in XML, to specify the following information:

- Opening Commands and Declarations
- The Root Element and Application Description
- Precompiling JSPs
- Application URLs
- Port Numbers and Security Constraints for Generated URLs
- Servlet Registration and Mapping
- URL Root for the AdClickThru Servlet
- Session Timeout
- Main Page and Error Page Mappings
- Declarations of Secure JSPs

For information on modifying parameters to enhance performance in the production environment, refer to the *Performance Tuning Guide*.

For more information about the XML elements that this section describes, refer to "web.xml Deployment Descriptor Elements" in *Developing WebLogic Server Applications.*

Opening Commands and Declarations

The deployment descriptor starts with an optional processing command:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
```

Then it declares its DTD:

```
<!DOCTYPE application PUBLIC '-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

The Root Element and Application Description

The root element of the deployment descriptor is named `<web-app>`, and a `</web-app>` tag closes it.

The following element syntax documents the purpose and scope of the web application, as specified in the *Java™ 2 Platform Enterprise Edition Specification, v1.3*:

```
<description>text description</description>
```

WebLogic Server does not use this element to identify the web application. Instead, it uses the module name that you specify in the enterprise application's `application.xml` file. For more information, refer to "Module Elements" on page 2-8.

Precompiling JSPs

The following element determines whether WebLogic Commerce Server compiles all JSPs in the web application when you start the server:

```
<context-param>
     <param-name>weblogic.jsp.precompile</param-name>
     <param-value>false</param-value>
</context-param>
```

By default, the element sets the value to `false`, which deactivates the JSP precompile option. With this option deactivated, the server starts quickly but must compile each new or modified JSP when you access it, causing a significant delay the first time you request a new or modified JSP.

When you activate the precompile option, the server startup process checks for new or modified JSPs in the web application and compiles them. Activating the precompile option can cause a significant delay in server startup if you have modified or added JSPs but avoids delays when you access a new or modified JSP for the first time.

Application URLs

The following element determines the URL that you use to request the WebLogic Commerce Server admin tools:

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

The following element determines the context for URLs that the Webflow mechanism generates:

```
<context-param>
   <param-name>WLCS_APPLICATION_URL</param-name>
   <param-value>/application/commercewf</param-value>
</context-param>
```

Port Numbers and Security Constraints for Generated URLs

The wlcs deployment descriptor contains several sets of elements to implement the J2EE declarative security model. This section describes the elements that configure the `createWebflowURL()` method to do the following:

- Generate Port Numbers for HTTP and HTTPS
- Determine Which Links Use HTTPS

Other elements, described later in this document, are responsible for the following:

- "Declarations of Secure JSPs" on page 2-20
- Declaring and mapping security roles, as described in "The weblogic.xml File" on page 2-22

Generate Port Numbers for HTTP and HTTPS

The following elements determine which port numbers the `createWebflowURL()` method encodes in the URLs that it generates:

```
<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>
```

If you want the wlcs web application to use the same port numbers as the WebLogic Server instance (for example, if you are using WebLogic Server as your HTTP server), either deactivate these elements by surrounding them in comment tags, `<!-- -->`, or change the values to match the listen port properties for the server. For more information, refer to "Changing the Listen Ports" on page 1-9.

If you want the wlcs web application to use different port numbers (for example, if you want to use the port numbers of a proxy server), change the values.

Determine Which Links Use HTTPS

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. Name specific resources by adding their name to the list. When naming specific pipelines and input processors, do not use the `.inputprocessor` or `.pipeline` extension.

Specify patterns for pipelines and input processors in the form of *`pattern`*`_*`. For example, to enable SSL for all requests to input processors whose names start with `profileeditcc_`, use the pattern `profileeditcc_*`.

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 2-20.

Servlet Registration and Mapping

The following element syntax registers a single servlet that the web application uses:

```
<servlet>
     <servlet-name>name that WebLogic Server uses when managing
     the servlet
     </servlet-name>
    <servlet-class>name of the class file that implements
    the servlet
    </servlet-class>
</servlet>
```

The e-Commerce sample web application includes several `<servlet>` elements to register several servlets.

The following element syntax defines a mapping between a servlet and a URL pattern:

```
   <servlet-mapping>
      <servlet-name>name that WebLogic Server uses when managing
     the servlet</servlet-name>
      <url-pattern>pattern used to resolve URLs</url-pattern>
   </servlet-mapping>
```

URL Root for the AdClickThru Servlet

The following element specifies the root for URLs that the AdClickThru servlet uses. Do not modify this element.

```
<context-param>
    <param-name>AD_CONTENT_BASE</param-name>
    <param-value>/wlcs</param-value>
</context-param>
```

Session Timeout

The <session-timeout> parameter determines how many minutes of inactivity the server will tolerate before ending the session. The WL_COMMERCE_HOME/server/webapps/wlcs/web.xml file sets the parameter to 15 minutes. You can modify this setting depending on your security needs.

```
<session-config>

   <session-timeout>15</session-timeout>

</session-config>
```

Main Page and Error Page Mappings

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

```
<welcome-file-list>

    <welcome-file>/index.jsp</welcome-file>

</welcome-file-list>
```

If you have named your main page something other than index.jsp, modify the value in your web application's deployment descriptor.

If a web application declares a main page, you can access the page by specifying the context root in the URL: http://host:port/context-name. You declare a web application's context root in the deployment descriptor for the enterprise application. For more information, refer to "Module Elements" on page 2-8.

The <error-page> elements use the following syntax to map error codes that the HTTP server returns to HTML files or JSPs:

```
<error-page>

   <error-code>number that the server returns</error-code>

   <location>pathname to page that displays an
   error message
  </location>

</error-page>
```

Declarations of Secure JSPs

The following <security-constraint> element syntax declares a collection of resources (JSPs) that only specific roles can access:

```
<security-constraint>

 <web-resource-collection>

    <web-resource-name>name of the resource collection</web-resource-name>

     <description>documentation of the collection's scope and purpose
     </description>

     <url-pattern>single URL pattern: use as many url-pattern elements as
     you need. One pattern/filename per element.
     </url-pattern>

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

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

    </web-resource-collection>


     <auth-constraint>

       <description>documentation of the authorization constraint</description>

       <role-name>name of declared security role: use as many role-name elements
       as you need. One role per element.
       </role-name>

    </auth-constraint>


     <user-data-constraint>

       <transport-guarantee>INTEGRAL or CONFIDENTIAL; see below for details
       </transport-guarantee>

    </user-data-constraint>

</security-constraint>
```

Use one of the following values in the `<transport-guarantee>` element:

- `INTEGRAL` to prevent the data from being changed while transmitting between the client and server.
- `CONFIDENTIAL` to prevent other entities from observing the contents of the transmission.

Note: Two files establish security roles for wlcs:

- The wlcsApp deployment descriptor, as described in "Global Security Role Elements" on page 2-9
- `weblogic.xml`, as described in "The weblogic.xml File" on page 2-22

The wlcs deployment descriptor declares a single role, CustomerRole, in the `<role-name>` element and several `<url-pattern>` elements. To give another role access to the resource collection, add `<role-name>` elements to the `<auth-constraint>` element.

You can add `<url-pattern>` elements to specify new directories or to specify specific files. Note that a pattern or filename must start with a `/` character (forward slash). For more information on specifying URL patterns, refer to the Java Servlet Specification v2.2.

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 "Determine Which Links Use HTTPS" on page 2-17.

The weblogic.xml File

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

For more information, refer to "weblogic.xml Deployment Descriptor Elements" in *Deploying WebLogic Server Applications*.

The `weblogic.xml` file contains sections to set the following properties:

- Opening Declaration
- The Root Element
- Security-Role Mappings
- JSP Deployment Options
- Naming Cookies

Opening Declaration

`weblogic.xml` starts by declaring its DTD:

```
<!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web
Application 6.0//EN" "http://www.bea.com/servers/wls600/dtd/
weblogic-web-jar.dtd">
```

The Root Element

The root element of `weblogic.xml` is named `<weblogic-web-app>`, and a `</weblogic-web-app>` tag closes it.

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 Commerce Server 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>
```

For more information on setting properties for J2EE security roles, refer to "Global Security Role Elements" on page 2-9.

JSP Deployment Options

Each web application includes a `<jsp-descriptor>` element to specify options for deploying JSPs. The `<jsp-descriptor>` element includes subelements to specify the following options:

- Saves the Java files that are generated as an intermediary step in the JSP compilation process. Unless this parameter is set to `true`, the intermediate Java files are deleted after they are compiled.

  ```
  <jsp-param>
   <param-name>keepgenerated</param-name>
   <param-value>false</param-value>
  </jsp-param>
  ```

- The frequency with which the server checks for modifications to JSPs. For more information, refer to the *Performance Tuning Guide*.

```
<jsp-param>
 <param-name>pageCheckSeconds</param-name>
 <param-value>300</param-value>
</jsp-param>
```

- Specifies the package into which all JSP pages are compiled.

```
<jsp-param>
 <param-name>packagePrefix</param-name>
 <param-value>jsp_compiled</param-value>
</jsp-param>
```

- The Java compiler that the web application uses. The WebLogic Server Administration Console specifies a Java compiler for the components in a WebLogic Server domain. You can override this property for a web application by removing the comment tags from the following element and specifying a pathname to a compiler in the `<param-value>` element:

```
<--
<jsp-param>
  <param-name>compileCommand</param-name>
  <param-value>compiler-pathname</param-value>
</jsp-param>
-->
```

- When set to `true`, debugging information is printed to the browser, the command prompt, and WebLogic Server log file.

```
<jsp-param>
 <param-name>verbose</param-name>
 <param-value>false</param-value>
</jsp-param>
```

For more information, refer to "weblogic.xml Deployment Descriptor Elements" in *Deploying WebLogic Server Applications*.

Naming Cookies

Each web application uses the following element to provide a unique name for the cookies that they set:

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

For more information, refer to "weblogic.xml Deployment Descriptor Elements" in *Deploying WebLogic Server Applications*.

3 Deploying Your Enterprise Application

Any Web site that uses Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server must be deployed as a web application or enterprise application that is part of a WebLogic Server domain.

This topic assumes that you are using the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server reference domain and enterprise application as a starting point for managing and developing your own enterprise application. It includes the following sections:

- Getting Started in a Development Environment
- Deploying Your EJBs or Web Applications
- Deploying in a Production Environment
- Dynamic Deployment

Before you continue, we recommend that you read the following:

- "Web and Enterprise Applications for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server" in the *Product Family Overview*.
- Chapter 1, "The Server Configuration."
- Chapter 2, "The Reference Domain."

Getting Started in a Development Environment

When you install Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server in your development environment, it is configured to activate the reference enterprise application and use the Cloudscape demonstration database that is populated with sample data. This default configuration is appropriate for demonstration purposes only.

Before your organization begins to develop its own enterprise application, complete the following deployment tasks in your development environment:

1. Place files under source control. For more information, refer to Place Files Under Source Control In addition, review "Tips for Developing Your Web Application."
2. Set up the database that you plan to use in your production environment.
3. Start the server, as described in Chapter 4, "Starting and Shutting Down the Server."
4. As you near the end of the development cycle, tune the wlcsServer performance for a testing or production environment. For information on Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server performance tuning, refer to the *Performance Tuning Guide*.

Any time during the development cycle, you can set up optional services for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server:

- SSL for secure communication. For more information, see "Configuring the SSL Protocol" under "Managing Security" in the *BEA WebLogic Server Administration Guide*.
- A security realm that uses an LDAP server. For more information, see Using WebLogic Realms under "Creating and Managing Users" in the *Guide to Building Personalized Applications* and "Configuring the LDAP Security Realm" under "Managing Security" in the *BEA WebLogic Server Administration Guide*.

- CyberCash, a credit-card validation service. For more information about CyberCash, refer to "Integration with CyberCash" under "Payment Services" in the *Guide to Managing Purchases and Processing Orders*. If you do not want to use CyberCash for credit card validation, refer to "What if I Don't Want to Use CyberCash for Credit Card Processing?" under "Payment Services."
- TAXWARE, a tax calculation service. For information about using TAXWARE, refer to "Integration with TAXWARE" under "Taxation Services" in the *Guide to Managing Purchases and Processing Orders*. If you do not want to use TAXWARE for tax calculation, refer to "What if I Don't Want to Use TAXWARE to Calculate My Taxes?" under "Taxation Services."
- A content management system. For more information, refer to "Using Third-Party Tools" under "Creating and Managing Content" in the *Guide to Building Personalized Applications*.

In addition, you can modify the flow of the order process to fit your business process. For more information, refer to "Tailoring Order Status to Your Business" under "Using the Order and Payment Management Pages" in the *Guide to Managing Purchases and Processing Orders*.

Place Files Under Source Control

Assuming that you develop an enterprise application based on the wlcsDomain, most of the files that developers modify while creating an enterprise application are located in two directory trees:

- `$WL_COMMERCE_HOME/config`, which contains wlcsDomain
- `$WL_COMMERCE_HOME/db`, which contains scripts that you use to create the WebLogic Commerce Server, WebLogic Personalization Server, and Campaign Manager for WebLogic schema in your environment

We recommend that you place both of these directory trees under source control in addition to the following WebLogic Commerce Server, WebLogic Personalization Server, and Campaign Manager for WebLogic properties files:

- `$WL_COMMERCE_HOME/pipeline.properties`
- `$WL_COMMERCE_HOME/webflow.properties`
- `$WL_COMMERCE_HOME/weblogiccommerce.properties`

When you start WebLogic Commerce Server, WebLogic Personalization Server, and Campaign Manager for WebLogic, it must have write access to the following directories:

- `$WL_COMMERCE_HOME`, where it places temp files and directories.
- `$WL_COMMERCE_HOME/config/wlcsDomain` and its subdirectories, where it writes log files and caching data.
- `$WL_COMMERCE_HOME/config/wlcsDomain/applications/defaultWebApp/WEB-INF`, where it maintains cached data and compiled classes for the default web application.
- `$WL_COMMERCE_HOME/config/wlcsDomain/applications/exampleportal/WEB-INF`, where it maintains cached data and compiled classes for the example portal web application.
- `$WL_COMMERCE_HOME/config/wlcsDomain/applications/tools/WEB-INF`, where it maintains cached data and compiled classes for the Administration Tools web application.
- `$WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcs/WEB-INF`, where it maintains cached data and compiled classes for the e-commerce web application.

To make modifications to a domain from the Administration Console, the corresponding `config.xml` file must be writable. For more information, refer to Chapter 2, “The Reference Domain.”

The Admin Tool web application writes its modifications to the WebLogic Commerce Server, WebLogic Personalization Server, and Campaign Manager for WebLogic database and does not write to any files in the domain.

Developing an enterprise application also modifies the following files and data structures:

- The database that contains WebLogic Commerce Server, WebLogic Personalization Server, and Campaign Manager for WebLogic data
- Data structures that you use to manage authentication and secure access, such as an LDAP server
- Files and data structures that your content management system uses

For guidelines on placing these files and data structures under source control, refer to the third-party vendor.

Tips for Developing Your Web Application

As you develop your own web applications, review the following reminders and suggestions:

- Using Webflow and the Flow Manager Servlet
- Providing Unique Names for Elements in the Webflow
- Working With Events
- Working with the Default Customer Profile

Using Webflow and the Flow Manager Servlet

If you use Webflow within your web application, you must register and provide URL mappings for the Flow Manager servlet. *Webflow* is a feature that controls the sequence of JSP files and the flow of data through input processors and Pipeline Components. It is dependent upon the Flow Manager servlet.

For more information about Webflow, refer to *Managing Presentation and Business Logic: Using Webflow and Pipeline*. For more information about the Flow Manager, refer to "Flow Manager" under "Foundation Classes and Utilities" in the *Guide to Building Personalized Applications*.

Listing 3-1 shows properties in an application's deployment descriptor (`web.xml` file) that register and provide URL mappings for the Flow Manager servlet. By default, the deployment descriptor for the sample e-commerce application includes these properties. If you use this deployment descriptor as a starting point for your own web application, you do not need to modify the properties that correspond to Listing 3-1. For more information, refer to "About the e-Commerce Deployment Descriptor" on page 2-13.

If you employ the Portal Framework, you cannot use Webflow to control the sequence of JSPs. For more information, refer to "Using Webflow Within a Portal" under "Advanced Portal Topics" in the *Guide to Creating Portals and Portlets*.

Listing 3-1 Flow Manager Properties in a Deployment Descriptor

```
<!-- Registers the flowManager servlet -->

<servlet>
<servlet-name>flowManager</servlet-name>
<servlet-class>com.beasys.commerce.foundation.flow.FlowManager</servlet-class>
</servlet>

<!-- Map all requests with the pattern "/application/*" to the FlowManager servlet
-->

<servlet-mapping>
<servlet-name>flowManager</servlet-name>
<url-pattern>/application/*</url-pattern>
</servlet-mapping>
```

Providing Unique Names for Elements in the Webflow

When working with Webflow, make sure that you provide unique names for links and buttons on JSP, input processors, and Pipelines. If you use the same name for two different elements, you will experience unwanted results. For more information, refer to "Using Webflow" under "Customizing Webflow and Pipelines" in *Managing Presentation and Business Logic: Using Webflow and Pipeline.*

Working With Events

If you use the Behavior Tracking and Events Service to track and respond to user actions, note that a user must be logged in (authenticated) before an event can be detected. Make sure that your modifications to the user registration and authentication features function properly before you modify and test features that rely on the Events Service, such as discounts, placeholders, and e-mail actions.

For more information, refer to "login.jsp Template" under "Customer Registration and Login Services" in the *Guide to Registering Customers* and the *Guide to Event and Behavior Tracking*.

Working with the Default Customer Profile

The the sample e-commerce web application provides a default customer profile and a CustomerProfileIP input processor to validate the customer registration form.

The form fields from the sample `login.jsp` refer to this customer profile as part of the authentication process. If you use these default form fields, you must make sure that your customers create profiles that are valid per the CustomerProfileIP. For more information, refer to "Customer Registration and Login Services" in the *Guide to Registering Customers*.

If you modify the customer profile and the CustomerProfileIP, you must make corresponding modifications to the CommitOrderPC Pipeline component, which validates customer profile information during the checkout process. Instead of modifying CommitOrderPC, you can write a new Pipeline component. For more information, refer to "CommitOrderPC" under "Order Summary and Confirmation Services" in the *Guide to Order Processing*.

Deploying Your EJBs or Web Applications

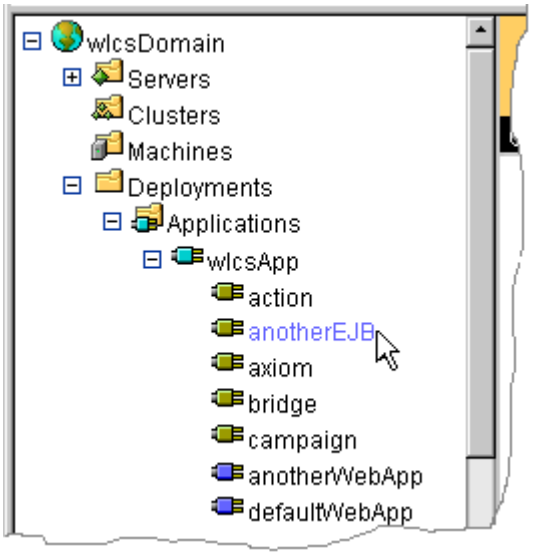
In the J2EE specification, EJBs and web applications are types of modules that you can add to an enterprise application. The process for deploying a J2EE module on WebLogic Server is similar for all module types.

If you develop EJBs to extend Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, or if you develop your own web application, do the following to add them to your enterprise application:

1. Copy the module into the following directory:
   `WL_COMMERCE_HOME/config/wlcsDomain/applications/wlscApp`
2. Edit the
   `WL_COMMERCE_HOME/config/wlcsDomain/applications/wlscApp/META-INF/application.xml` file, and place a reference to the new module there. For information about creating references to modules, refer to "Module Elements" on page 2-8.
3. If the server is running, restart it.
4. Start the WebLogic Administration Console for your domain. For more information, refer to "Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain" on page 1-7.

5. In the Administration Console, in the left pane, click Deployments → Applications → wlcsApp.
6. Under wlcsApp, click the EJB or web application that you added to `application.xml`. (See Figure 3-1.)

Figure 3-1 Click the EJB or Web Application That You Added

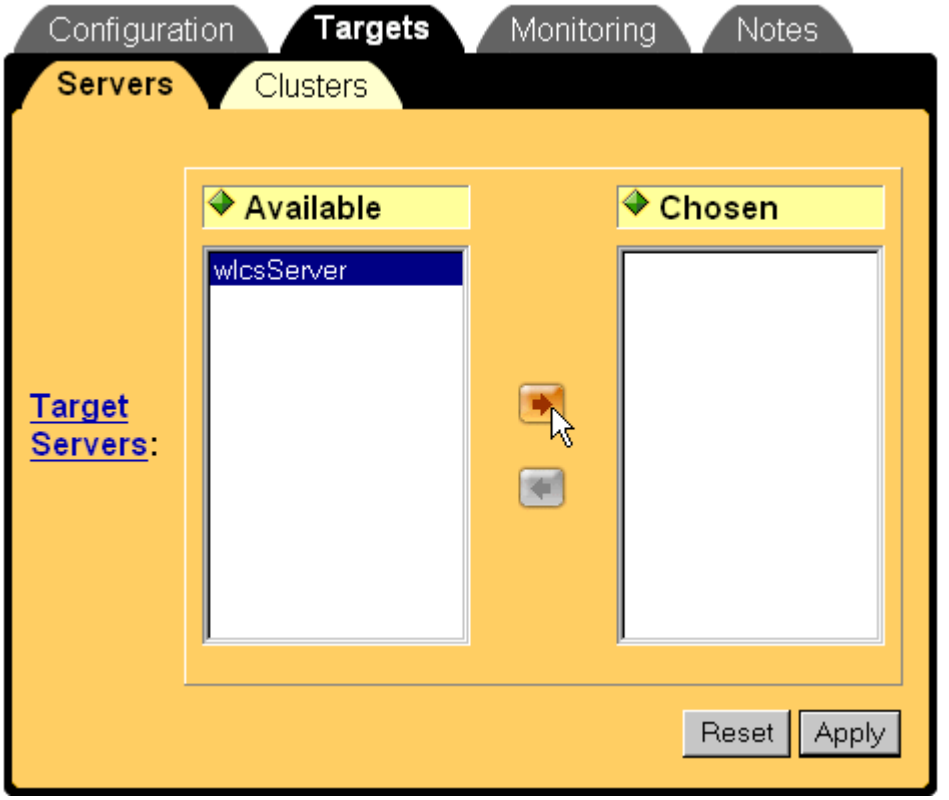


The right pane of the Administration Console displays the module deployment properties.

7. In the right pane, click the Targets tab. Then click the Servers subtab.

8. On the Servers tab, move wlcsServer from the Available list to the Chosen list. (See Figure 3-2.)
9. Click Apply.

Figure 3-2 Deploy the Module on wlcsServer



Deploying in a Production Environment

When you are ready to make your enterprise application available in a production environment, do the following:

1. Install Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server in the production environment.
2. Set up your database in the production environment. For information on exporting Oracle data objects from a development environment to a production environment, refer to Chapter 6, "Migrating Oracle Data Objects."
3. Update Properties Files for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.
4. Update wlcsDomain.

Note: For information on deploying your enterprise application on Managed Servers in a cluster, refer to "Starting WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Classes" on page 4-8. If you need to make further performance tuning adjustments in the production environment, refer to the *Performance Tuning Guide*.

Update Properties Files for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server

As part of developing your enterprise application, developers may modify the following files, which help to control the presentation and business processing logic in the e-commerce web application:

- `WL_COMMERCE_HOME/pipeline.properties`
- `WL_COMMERCE_HOME/webflow.properties`

In addition, you or another administrator may have made modifications to the `WL_COMMERCE_HOME/weblogiccommerce.properties` file in the development environment that you want to make available in the production environment. For example, the `weblogiccommerce.properties` file determines which database Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server uses.

Copy all three files into the `WL_COMMERCE_HOME` directory of the production environment.

Update wlcsDomain

To make your enterprise application available to the wlcsDomain in the production environment, replace the `WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp` directory tree in the production environment with the `wlcsApp` directory tree that your developers modified in the development environment.

Then start the server in the production environment.

You can also use the Java `jar` command to compress the enterprise application directory tree into a single Enterprise Application Archive (EAR) file. Deploying a single file on your production server can facilitate Web site administration by reducing the number of files you must keep track of.

To modify anything inside the EAR file, you must uncompress the file, make your modifications, and then re-compress the file.

Instead of compressing the entire enterprise application, you can also use the Java `jar` command to compress your web applications into Web Application Archive (WAR) files. If you use WAR files, you must update the enterprise application's `application.xml` file to refer to the web application as a WAR file instead of as an expanded directory.

Dynamic Deployment

Dynamic deployment is modifying or adding a managed object without restarting the server. You can modify any existing JSP, HTML, image file in an active domain and Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server will deploy the modification per the intervals that you have established in the domain and web application properties.

For example, you set the JSP page-check interval for the e-commerce web application to 5 minutes. A developer added an ad banner to a catalog JSP, `wlcsApp/wlcs/catalog/browse.jsp`, and requests that you update the JSP in production environment. You then copy the modified `browse.jsp` into the `wlcsApp/wlcs/catalog` directory in the production environment. Depending on when WebLogic Server last checked for modified JSPs, it could take up to 5 minutes for WebLogic Server to display the modified JSP.

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server does not dynamically deploy modifications of the following types of files:

- EJBs or class files
- WARs or EARs
- `weblogiccommerce.properties`
- Deployment descriptors and the configuration file for the domain

4 Starting and Shutting Down the Server

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provide two files, `set-environment` and `StartCommerce`, that start the server and activate the wlcsDomain (and all web applications in the domain). We recommend that you use these files to start any web applications in the wlcsDomain.

In addition, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server provide a `StopCommerce` script, which gracefully shuts down the server and some third-party integrations.

This topic includes the following sections:

- Starting the Server and wlcsDomain on UNIX
- Starting the Server and wlcsDomain on Windows
- Startup Confirmation
- Setting Environment Variables
- Starting an HTTP Server for TAXWARE
- Starting WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Classes
- Shutting Down the Server

Starting the Server and wlcsDomain on UNIX

From a Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host, enter the following command where `WL_COMMERCE_HOME` is the directory into which you installed Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server:
`WL_COMMERCE_HOME/StartCommerce.sh`

The `StartCommerce.sh` calls `set-environment.sh` to set environment variables. Then it passes to the JVM the class name and parameters that start WebLogic Server, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, and activate the wlcsDomain.

For information on starting the server without using `StartCommerce.sh` (for example, if you have modified the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server environment), refer to the following sections:

- Setting Environment Variables
- Starting an HTTP Server for TAXWARE
- Starting WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Classes

Starting the Server and wlcsDomain on Windows

From a Windows Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host, do one of the following:

- Click Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → Start WebLogic Commerce Server
- From a command prompt, enter the following command:
  `%WL_COMMERCE_HOME%\StartCommerce.bat`

The Start WebLogic Commerce Server command on the Start menu is a shortcut to `StartCommerce.bat`. The `StartCommerce.bat` calls `set-environment.bat` to set environment variables. Then it passes to the JVM the class name and parameters that start WebLogic Server, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, and activate the wlcsDomain.

For information on starting the server without using the Start command or `StartCommerce.bat` (for example, if you have modified the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server environment), refer to the following sections:

- Setting Environment Variables
- Starting an HTTP Server for TAXWARE
- Starting WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Classes

Startup Confirmation

When you issue a startup command, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server displays messages in the shell that contains the server process. The following three messages indicate that the server started successfully:

```
< DATE > <Notice> <WebLogicServer> <WebLogic Server started>
```

```
< DATE > <Notice> <WebLogicServer> <SSLListenThread listening on port NUMBER>
```

```
< DATE > <Notice> <WebLogicServer> <ListenThread listening on port NUMBER>
```

For information about changing the number and type of messages that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server displays in the shell, refer to "Viewing and Modifying Properties in the WebLogic Server Administration Console" on page 1-6.

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.

- `WL_COMMERCE_HOME=`{ Directory into which you installed Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. }
- `JDK_HOME=`{ Directory into which you installed the JDK that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server require. }
- `WLCS_ORACLE_HOME=`{ If you are using Oracle, set this variable to the directory in which you installed the Oracle client software. }
- `BEA_HOME=`{ Directory that contains BEA license files. }
- `WEBLOGIC_HOME=`{ Directory into which you installed WebLogic Server. }
- `SQLPATH=` { If you are using an Oracle database, set this variable to the directories that contain Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server scripts for creating tablespaces and schemas. }
  For example,
  ```
  D:\bea\WebLogicCommerceServer4.0\db\oracle;D:\bea\WebLogicCommerceServer4.0\db\oracle\event;D:\bea\WebLogicCommerceServer4.0\db\oracle\migration;D:\bea\WebLogicCommerceServer4.0\db\oracle\wlcs;D:\bea\WebLogicCommerceServer4.0\db\oracle\wlps
  ```

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%\eval\cloudscape\lib\cloudscape.jar;%WEBLOGIC_HOME%\eval\cloudscape\lib\tools.jar;%WEBLOGIC_HOME%\eval\cloudscape\lib\client.jar`
- The pathname of the `tools.jar` file in the JDK that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server require. 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;%WEBLOGIC_HOME%\ext\weblogic-tags.jar`
- The classes required for the WebLogic Personalization Server run-time environment.
  For example,
  `%WL_COMMERCE_HOME%\lib\jdom.jar;%WL_COMMERCE_HOME%\lib\mail.jar;%WL_COMMERCE_HOME%\classes;%WL_COMMERCE_HOME%\lib\wlcsparsers.jar;%WL_COMMERCE_HOME%\license`
- The classes required for the WebLogic Commerce Server run-time environment.
  For example,
  `%WLPS_CLASSPATH%;%WL_COMMERCE_HOME%\eval\common\Taxware\classes`

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`
- The directory that contains the CyberCash binary files.
  For example, `%WL_COMMERCE_HOME%\eval\win32\CyberCash\bin`
- The directory that contains the TAXWARE binary files.
  For example, `%WL_COMMERCE_HOME%\eval\win32\Taxware\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\oci816_8;%WLCS_ORACLE_HOME%\bin`

Starting an HTTP Server for TAXWARE

If you use the WebLogic Commerce Server integration with TAXWARE, you must start a small HTTP server before you start WebLogic Commerce Server. The HTTP server supports HTTP requests to and from the TAXWARE integration.

The following statements from the StartCommerce.bat script stop the HTTP server if it is already running, then it starts the HTTP server and verifies the startup:

```
REM ----------- Stop the netservice http helper if it's running -----------

%JDK_HOME%\bin\java -classpath %CLASSPATH%
com.beasys.commerce.netservice.http.util.Shutdown -host 127.0.0.1 -port 6519
-quiet



REM ----------- Start the netservice http helper -----------

start /b %JDK_HOME%\bin\java %JAVA_VM% -classpath %CLASSPATH%
com.beasys.commerce.netservice.http.server.SimpleHTTPServer
```

```
REM
REM sleep for 2 seconds to allow the server to startup
REM

%JDK_HOME%\bin\java -classpath %CLASSPATH%
com.beasys.commerce.netservice.http.util.Sleep 2

REM
REM ping the server to make sure it started
REM

%JDK_HOME%\bin\java -classpath %CLASSPATH%
com.beasys.commerce.netservice.http.util.Ping -host 127.0.0.1 -port 6519 -quiet
```

Starting WebLogic Server with Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Classes

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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server classes:

- `-hotspot` (Windows only)

  Invokes the HotSpot Client VM, which Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server requires for Windows. For more information, see "Use the HotSpot Virtual Machine" in the *Performance Tuning Guide*.

- `-server` (UNIX only)

  Invokes the HotSpot Server VM, which Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server requires for UNIX. For more information, see "Use the HotSpot Virtual Machine" in the *Performance Tuning Guide*.

- `-Xms128m -Xmx128m`

  Specifies the initial and maximum values for Java heap memory. These values assigned to these parameters can dramatically affect the performance of your 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 4-6.

- If you use the Cloudscape database, include the following argument to provide the database location:
  `-Dcloudscape.system.home=%WL_COMMERCE_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/myWlcsDomain`, then use the following argument:
`-Dweblogic.Domain=myWlcsDomain`

- `-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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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=`*`WL_COMMERCE_HOME`*`/weblogiccommerce.properties`
  `-Dpipeline.properties=`*`WL_COMMERCE_HOME`*`/pipeline.properties`
  `-Dwebflow.properties=`*`WL_COMMERCE_HOME`*`/webflow.properties`

- `weblogic.Server`

  Where *`WL_COMMERCE_HOME`* is the directory into which you installed Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. These arguments load Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server classes.

For example, the StartCommerce script for an Administration Server on Windows issues the following command:

```
%JDK_HOME%\bin\java -hotspot -Xms128m -Xmx128m -classpath %CLASSPATH%
-Dcloudscape.system.home=%WL_COMMERCE_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=%WL_COMMERCE_HOME%/weblogiccommerce.properties
-Dpipeline.properties=%WL_COMMERCE_HOME%/pipeline.properties
-Dwebflow.properties=%WL_COMMERCE_HOME%/webflow.properties weblogic.Server
```

Shutting Down the Server

To shut down the server and some of the third-party integration processes, use the `StopCommerce` script, which is located in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server installation directory.

To use the `StopCommerce` script on UNIX, open a shell and enter the following command:

```
WL_COMMERCE_HOME/StopCommerce.sh
```

To use the `StopCommerce` script on Windows, enter the following command from a DOS prompt or equivalent command prompt:

```
WL_COMMERCE_HOME/StopCommerce.bat
```

where `WL_COMMERCE_HOME` is the directory in which you installed Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server.

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, the WebLogic Server Administration Console can shut down Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, but it leaves some third-party processes running.

Part II Deploying the Data Repository

5 Setting Up Oracle for New Installations

To use an Oracle database with an installation of Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, complete the following tasks:

Step 1: Create Tablespaces for Oracle

Step 2: Install the Oracle Client Software

Step 3: Create Oracle User Accounts

Step 4: Create the Schema for Oracle

Step 5: Rebuild Indexes

Step 6: Configure Properties Files and Environment Variables for Oracle

Step 7: Load Additional Sample Data

Step 1: Create Tablespaces for Oracle

To separate its data from the Oracle data dictionary and from data that other applications use, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server uses the following tablespaces:

- WLCS_DATA, which contains tables for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server
- WLCS_INDEX, which contains indexes for Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server
- WLCS_EVENT_DATA, which contains tables for Behavior Tracking (Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server need this tablespace only if you use the Behavior Tracking feature.)

For optimal performance, we recommend that you reduce I/O contentions by placing each tablespace on its own physical disk drive.

Creating WLCS_DATA and WLCS_INDEX

Note: In this document, `WL_COMMERCE_HOME` refers to the directory into which you installed WebLogic Commerce Server and/or WebLogic Personalization Server.

1. On the server that hosts the Oracle database, start SQL*Plus and log in as SYSTEM (or another user with create-tablespace privileges). You must complete this procedure on the local host; you cannot successfully create tablespaces from a remote session.
2. Make a backup copy of `WL_COMMERCE_HOME/db/oracle/8.1.6/admin/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 directory path structures. For example, on a UNIX workstation two disks are mounted as `/usr1` and `/usr2` and the Oracle SID is `PROD`. We recommend the following pathnames:
   ```
   DEFINE DATA_PATHNAME=/usr1/oracle/oradata/PROD
   DEFINE INDEX_PATHNAME=/usr2/oracle/oradata/PROD
   ```

4. In the SQL*Plus session you started in step 1, enter the following command:
   ```
   @ WL_COMMERCE_HOME/db/oracle/8.1.6/admin/create_tablespaces.sql
   ```

Creating WLCS_EVENT_DATA

Creating the WLCS_EVENT_DATA tablespace is necessary only if you are using the Behavior Tracking feature.

1. On the server that hosts the Oracle database, start SQL*Plus and log in as SYSTEM (or another user with create-tablespace privileges). You must complete this procedure on the local host; you cannot successfully create tablespaces from a remote session.

2. Make a backup copy of
   `WL_COMMERCE_HOME/db/oracle/8.1.6/event/create_event_tablespaces.sql`

   In a text editor, open `create_event_tablespaces.sql` and modify the pathname for the `EVT_DATA_PATHNAME` variable to match your own directory path structure.

3. In the SQL*Plus session you started in step 1, enter the following command:
   ```
   @ WL_COMMERCE_HOME/db/oracle/8.1.6/event/create_event_tablespaces.sql
   ```

Step 2: Install the Oracle Client Software

On the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host, use the Oracle installation program to install Oracle client software.

For information on the Oracle release that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server supports for your platform type, refer to Supported Platforms in the *Installation Guide*.

During the installation process, use the Oracle Net8 Assistant application to set up a new TNS service. The Net8 Assistant prompts you for the name or IP address of the computer that hosts the Oracle server software, the port number Oracle is listening on (usually 1521), and the SID name for the database. Test the connection with the database username and password that your database administrator assigned to you.

If you are using the WebLogic jDriver to connect to the Oracle database, refer to "Installing and Using WebLogic jDriver for Oracle" on the WebLogic documentation site for information on setting up the appropriate jDriver.

Step 3: Create Oracle User Accounts

To simplify administration, we recommend that you create a user account exclusively for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server schema. The servers include a script that creates a user account named WEBLOGIC and specifies a default tablespace for the account. Although you can modify the script to change the account name, we recommend that you do not; several other scripts assume that the account is named WEBLOGIC. If you change the account name, you must review each script for the use of the WEBLOGIC account name and modify it.

To create a user account:

1. From the host on which you installed the Oracle client software, start SQL*Plus and log in as SYSTEM (or any user with create-user privileges).
2. Enter the following command:
   `@ WL_COMMERCE_HOME/db/oracle/8.1.6/admin/create_users.sql`

Step 4: Create the Schema for Oracle

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server include a script, `create_all.sql`, that calls a series of other scripts to create the WebLogic Commerce Server and WebLogic Personalization Server schema and install sample data. You can modify the script to create the database without loading sample data.

This step includes the following tasks:

- Prevent Sample Data from Loading (optional)
- Run create_all.sql

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*.

Prevent Sample Data from Loading (optional)

By default, `create_all.sql` calls other scripts to load sample catalog, customer, and user data. The sample Web applications use this data to demonstrate Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server features.

If you do not want to use this sample data, you can create the database without inserting sample data by modifying `create_all.sql` as described in this section.

To prevent sample data from loading, do the following:

Note: The sample Web applications do not function properly without this sample data. For example, the example portal Web application adds new users to the AcmeUsers group. If you do not load sample data, then the AcmeUsers group does not exist and the example portal fails to create new users. Do not follow this procedure if you are setting up an environment in which you want to run the sample Web applications.

1. Create a backup copy of `WL_COMMERCE_HOME/db/oracle/8.1.6/create_all.sql`
2. Open `create_all.sql` in a text editor.
3. Deactivate the statements that insert sample data by adding remark comments (`--`) to the front of the line. Listing 5-1 shows the statements that insert sample data as deactivated and in bold font.

Listing 5-1 Deactivated Insert Statements

```
...

-- SET HEADING OFF
-- select '************** insert_wlps_sample_data.sql' from dual;
-- SET HEADING ON
-- @insert_wlps_sample_data.sql

SET HEADING OFF
select '************** insert_wlcs.sql' from dual;
SET HEADING ON
@insert_wlcs.sql

-- SET HEADING OFF
-- select '************** insert_wlcs_sample_data.sql' from dual;
-- SET HEADING ON
-- @insert_wlcs_sample_data.sql

-- SET HEADING OFF
-- select '************** insert_wlcs_sample_catalog.sql' from
dual;
-- SET HEADING ON
-- @insert_wlcs_sample_catalog.sql

-- SET HEADING OFF
-- select '************** insert_wlcs_sample_customer.sql' from
dual;
-- SET HEADING ON
-- @insert_wlcs_sample_customer.sql
```

4. Save your modifications to `create_all.sql`.

Run create_all.sql

To create the WebLogic Commerce Server and WebLogic Personalization Server schema, do the following:

1. On the Oracle client host, log in to the WEBLOGIC user account by entering the following command from SQL*Plus:
   `connect WEBLOGIC/WEBLOGIC@ORACLE_SID`
2. Enter the following command:
   `@ WL_COMMERCE_HOME/db/oracle/8.1.6/create_all.sql`

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 Creating 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: Rebuild Indexes

The `create_all.sql` places the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server index in the default tablespace (`WLCS_DATA`). To locate and rebuild the index; in the `WLCS_INDEX` tablespace, do the following:

1. From the Oracle client host, log in to the WEBLOGIC user account by entering the following command from SQL*Plus:
   `connect WEBLOGIC/WEBLOGIC@ORACLE_SID`
2. Enter the following command:
   `@ WL_COMMERCE_HOME/db/oracle/8.1.6/admin/rebuild_indexes.sql`

Step 6: Configure Properties Files and Environment Variables for Oracle

To configure properties files and environment variables for Oracle, complete the following tasks:

- Set Up the JDBC Connection Pool
- Edit the weblogiccommerce.properties File
- Update Environment Variables for the Server

Set Up the JDBC Connection Pool

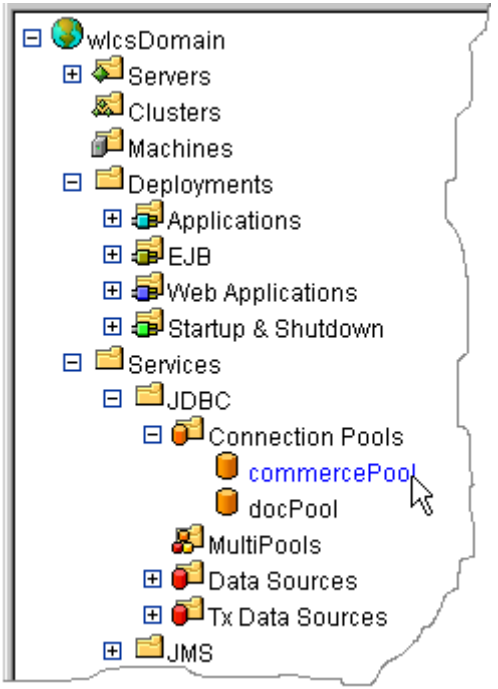
To set up the JDBC connection pool for Oracle, do the following:

1. Start the WebLogic Administration Console for your domain. For more information, refer to “Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain” on page 1-7.

2. In the left pane of the Administration Console, click Services → JDBC → Connection Pools → commercePool. (See Figure 5-1.)

   The right pane of the Administration Console displays the commercePool properties.

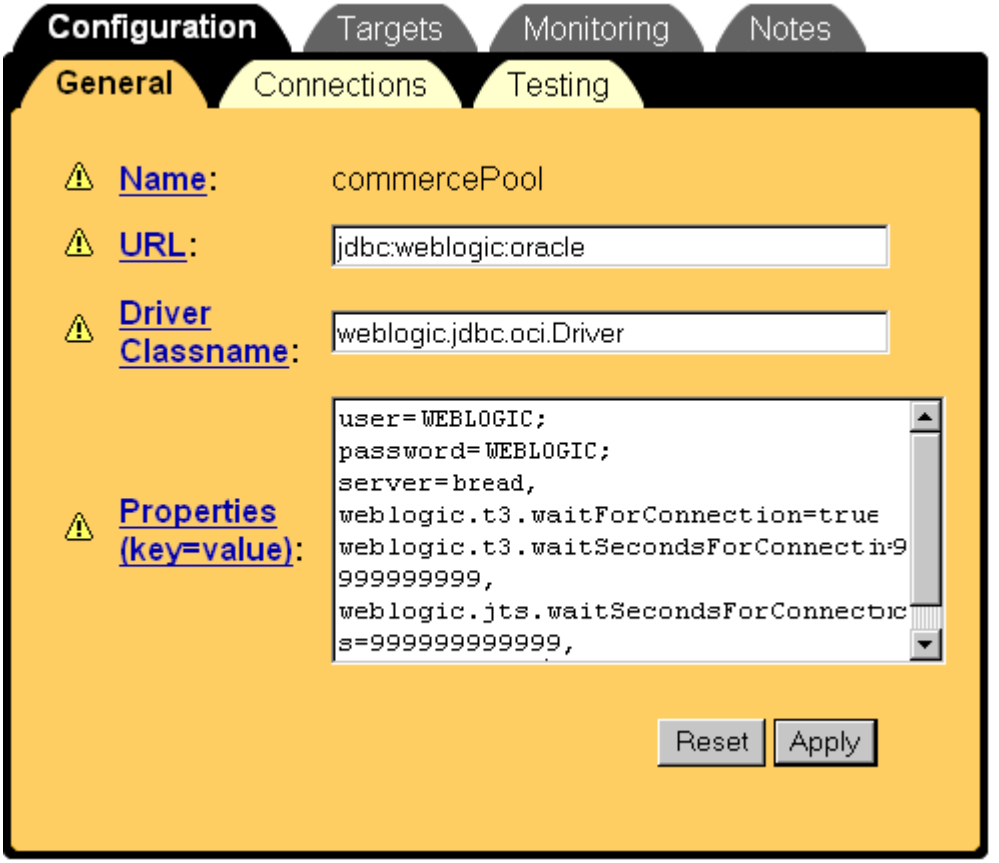
Figure 5-1 Click commercePool



3. On the commercePool page, click the Configuration tab. Then click the General subtab.

4. On the General subtab, in the URL box, enter the following path:
   `jdbc:weblogic:oracle`
5. In the Driver Classname box, enter the following driver name:
   `weblogic.jdbc.oci.Driver`
6. In the Properties box, enter the following properties (See Figure 5-2.):
   ```
   user=@ORACLE_USER@;
   password=@ORACLE_PASSWORD@;
   server=@ORACLE_NET_SERVICE_NAME@,
   weblogic.t3.waitForConnection=true;
   weblogic.t3.waitSecondsForConnection=999999999999,
   weblogic.jts.waitSecondsForConnectionSecs=999999999999,
   verbose=false
   ```
   where *@ORACLE_USER@* is the name of the Oracle user account, *@ORACLE_PASSWORD@* is the user account's password, and *@ORACLE_NET_SERVICE_NAME@* is the name of the Oracle net service.
7. On the General subtab, click Apply.

Figure 5-2 General Pool Properties



8. Click the Connections subtab.

9. Enter values for the properties as follows:
   - Initial Capacity = 1
   - Maximum Capacity = 20
   - Capacity Increment = 1
   - Login Delay Seconds = 0
   - Refresh Period = 5
   - select Allow Shrinking
   - Shrink Period = 15
10. Click Apply.
11. Click the Testing subtab and do the following:
    - In Test Table Name, enter `WLCS_IS_ALIVE`
    - Select Test Reserved Connections
    - Clear Test Released Connections
12. Click Apply.

    You must restart the server to deploy the modifications. However, we recommend that you complete the remaining procedures in this topic before you restart the server.

Edit the weblogiccommerce.properties File

If you are using the RDBMS security realm, you must change the `RDBMSRealm` settings in the `weblogiccommerce.properties` file 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.

For information on the Oracle release that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server supports for your platform type, refer to Supported Platforms in the *Installation Guide*.

For example, Listing 5-2 shows a `weblogiccommerce.properties` file that specifies the WebLogic jDriver.

To change these settings, you must do **all** of the following:

1. Remove the comment tags from the lines that specify the database type you are using.
2. Change any placeholders in the lines. For example, you must change the string `@ORACLE_SERVER@` to the name of the Oracle server that hosts the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server database.
3. Deactivate any other lines in this section by adding comment tags to the beginning of each line.
4. Make sure that the value for `commerce.usermgmt.RDBMSRealm.dbUrl` matches the value for the `commerce.jdbc.pool.url` property (also located in `weblogiccommerce.properties`), which is used to establish JDBC connection pools.

   Also verify that the connection pool properties in the WebLogic Server Administration Console specify the same database type as these settings for the RDBMS realm. For more information, refer to "Set Up the JDBC Connection Pool" on page 5-10.

Listing 5-2 Settings for the RDBMS Realm

```
# RDBMS Realm

#------Oracle thin Driver---------#

#commerce.usermgmt.RDBMSRealm.driver=oracle.jdbc.driver.OracleDriver
#commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:oracle:thin:ENTERPRISE:1205:COMMERCE
#commerce.usermgmt.RDBMSRealm.dbUser=@ORACLE_USER@
#commerce.usermgmt.RDBMSRealm.dbPassword=@ORACLE_PASSWORD@

#------WebLogic jDriver for Oracle 8.1.5---------#

commerce.usermgmt.RDBMSRealm.driver=weblogic.jdbc.oci.Driver
commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:weblogic:oracle
commerce.usermgmt.RDBMSRealm.dbServer=bread
commerce.usermgmt.RDBMSRealm.dbUser=WEBLOGIC
commerce.usermgmt.RDBMSRealm.dbPassword=WEBLOGIC

#--------------Cloudscape------------------#

#commerce.usermgmt.RDBMSRealm.driver=COM.cloudscape.core.JDBCDriver
#commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:cloudscape:Commerce;create=true;autoco
```

```
mmit=false
#commerce.usermgmt.RDBMSRealm.dbUser=none
#commerce.usermgmt.RDBMSRealm.dbPassword=none
```

Update Environment Variables for the Server

To update environment variables, do the following:

1. In a text editor, open `%WL_COMMERCE_HOME%\bin\`*`platform-type`*`\set-environment.bat` (Windows) or `WL_COMMERCE_HOME/bin/`*`platform-type`*`/set-environment.sh` (UNIX).

A subsequent section describes in detail each of the remaining steps.

2. Set `@ORACLE_HOME@`.
3. Specify the database.
4. Set variables for Oracle drivers.

Set @ORACLE_HOME@

After opening `set-environment`, find the line `WLCS_ORACLE_HOME=@ORACLE_HOME@` and substitute `@ORACLE_HOME@` with the absolute pathnames of the directory in which you installed the Oracle client software.

For example, Listing 5-3 shows a Windows environment in which the Oracle client software is installed in `D:\oracle`.

Listing 5-3 Set @ORACLE_HOME@

```
SET WL_COMMERCE_HOME=D:\WebLogicCommerce3.5
SET JDK_HOME=D:\jdk1.3

SET WLCS_ORACLE_HOME=D:\oracle

SET WEBLOGIC_HOME=D:\weblogic

REM -- Add WebLogic, CyberCash, and Taxware bin directories to the path --
```

```
SET PATH=%PATH%;%WEBLOGIC_HOME%\bin;%WL_COMMERCE_HOME%\eval\win32\CyberCash\bin;
%WL_COMMERCE_HOME%\eval\win32\Taxware\bin
```

The WLCS_ORACLE_HOME variable adds Oracle-client Java classes and libraries to the Java classpath and to the system path.

Specify the Database

Under the section titled
-------- Specify which database to use --------,
activate the line that sets the DATABASE variable to the database type that you are using and deactivate any other line. For example, Listing 5-4 shows a Windows environment that uses the WebLogic jDriver.

Listing 5-4 Specify the Database

```
REM ----------- Specify which database to use -----------

REM SET DATABASE=CLOUDSCAPE

REM DATABASE=ORACLE

SET DATABASE=ORACLE_OCI_815


if %DATABASE% EQU CLOUDSCAPE GOTO CLOUDSCAPE
if %DATABASE% EQU ORACLE GOTO ORACLE
if %DATABASE% EQU ORACLE_OCI_815 GOTO ORACLE_OCI_815

:CLOUDSCAPE

REM ----------- CLOUDSCAPE classes -----------
SET
DB_CLASSPATH=%WEBLOGIC_HOME%\eval\cloudscape\lib\cloudscape.jar;%WEBLOGIC_HOME%
\eval\cloudscape\lib\tools.jar;%WEBLOGIC_HOME%\eval\cloudscape\lib\client.jar

GOTO continue
```

Set Variables for Oracle Drivers

In `set-environment`, under the `:ORACLE_OCI_815` section, verify that the `PATH` variable correctly identifies the location of the WebLogic jDriver and Oracle client libraries.

The following example from a Windows installation adds the WebLogic jDriver and Oracle client libraries to the system path:

```
:ORACLE_OCI_815

REM ----------- WLS ORACLE 815 classes -----------

SET DB_CLASSPATH=%BEA_HOME%

SET PATH=%PATH%;%WEBLOGIC_HOME%\bin\oci816_8;%WLCS_ORACLE_HOME%\bin
```

Step 7: Load Additional Sample Data

You can load additional sample data that demonstrates ad placeholders and features that are available only with Campaign Manager for WebLogic. If you do not use Campaign Manager for WebLogic, you can prevent campaign-specific data from loading by modifying the `loadSampleData` script.

This section describes the following tasks:

- Modifying loadSampleData
- Running loadSampleData

Modifying loadSampleData

The `loadSampleData` script is located at the following pathname:

`WL_COMMERCE_HOME\bin\`*`platform-type`*`\loadSampleData.bat` (Windows)
`WL_COMMERCE_HOME/bin/`*`platform-type`*`/loadSampleData.sh` (UNIX)

Note: The remainder of this section uses Windows syntax and filenames.

It calls the following scripts to load data for features that are available in Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server:

- `loaddocs.bat`, which loads documents that the example portal web application displays.
- `loadads.bat`, which loads ads to demonstrate ad placeholders in the example portal application and the e-commerce sample web application.
- `loadSampleDiscounts.bat`, which loads discounts to demonstrate the Discount Service.

It calls the following scripts to load data for features that are available only in Campaign Manager for WebLogic (if you do not use Campaign Manager for WebLogic, you can remove or deactivate the lines that call these scripts):

- `loadrules.bat`, which loads conditions and customer segments to demonstrate Campaign Manager for WebLogic features.
- `loadSampleCampaigns.bat`, which loads a sample campaign.

Running loadSampleData

To load additional sample data, do the following:

1. From a DOS prompt on the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host, enter the following command:
   `%WL_COMMERCE_HOME%\StartCommerce.bat`
2. Open another command shell and enter the following command:
   `%WL_COMMERCE_HOME%\bin\`*`platform-type`*`\loadSampleData.bat`
3. Stop and restart Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. For information on stopping the server, refer to “Shutting Down the Server” on page 4-11.

6 Migrating Oracle Data Objects

Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server includes a set of SQL scripts that create Oracle tables and other data objects in your testing, staging, or production environment. After creating the data objects, you can use Oracle commands to export data from the source environment and import it into a destination environment.

The Oracle commands export and import all Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server data including any document metadata that is part of the default content management implementation. The commands do not move the following:

- JSP files or other configuration files.
- Documents and document metadata that is maintained by a third-party content-management tool, such as Documentum or Interwoven.

To export Oracle data objects to a destination environment, complete the following tasks:

Step 1: Create the Destination Environment

Step 2: Review Parameter Files

Step 4: Delete Sample Data (Optional)

Step 5: Remove Orphaned Records

Step 6: Export the Data

Step 7: Stop the Server in the Destination Environment

Step 8: Import the Data

Step 9: Start the Server in the Destination Environment

Step 1: Create the Destination Environment

To create the destination environment, follow the procedures in Chapter 5, "Setting Up Oracle for New Installations."

For more information about Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server database schemas, refer to the following documents:

- "The WebLogic Personalization Server Database Schema" in the *Guide to Building Personalized Applications*
- "The Product Catalog Schema" in the *Guide to Building a Product Catalog*
- "The Order Processing Database Schema" in the *Guide to Managing Purchases and Processing Orders*

Step 2: Review Parameter Files

Complete this step only if you use a third-party solution (such as Interwoven or Documentum) for content management.

If your Oracle user account contains data objects that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server does not create and maintain (such as Interwoven and Documentum objects), you must use parameter files to establish user identity and determine the tables for which you want to export and import data.

Reviewing the parameter files involves the following tasks:

- Review the List of Tables
- Add FROMUSER to the Import Parameter File

For more information about the Oracle export and import commands and about the function of the parameters in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server parameter files, see the following Oracle document:
http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/server.817/a76955/toc.htm.

Note: To access articles on technet.oracle.com, you must register as a member of Oracle Technology Network. For more information, refer to http://technet.oracle.com.

Review the List of Tables

Complete this procedure only if you use a third-party solution (such as Interwoven or Documentum) for content management.

Note: In this document, `WL_COMMERCE_HOME` refers to the directory into which you installed WebLogic Commerce Server and/or WebLogic Personalization Server.

Review the list of tables in the parameter files and, if necessary, modify them to match your environment:

1. Save backup copies of the following files:
   `WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/wlcs_exp.par`

   `WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/wlcs_imp.par`
2. Open both files in a text editor.
3. If you are using Interwoven or Documentum for content management, prevent the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server scripts from exporting and importing data into content management tables by removing the following table names:
   - WLCS_DOCUMENT
   - WLCS_DOCUMENT_METADATA

   Note: If you are using the reference implementation (BulkLoader) for content management, you must leave these table names in the files.
4. If you added or removed tables from the default database schemas, add or remove the corresponding table names in the parameter files.
5. Save your modifications to the parameter files.

Add FROMUSER to the Import Parameter File

Complete this procedure only if you use a third-party solution (such as Interwoven or Documentum) for content management.

To specify the user account that owns the schema you want to import, do the following:

1. Make a backup copy of the following file:
   `WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/wlcs_imp.par`
2. Open `wlcs_imp.par` in a text editor.
3. For the `FROMUSER` entry, specify the name of the user who owned the schema in the source environment. For example, in your development environment, the owner of the Oracle schema is `WEBLOGIC`. In `wlcs_imp.par`, you modify the first line as follows:
   `FROMUSER=WEBLOGIC`

Step 4: Delete Sample Data (Optional)

Caution: Before you delete sample data, **back up your database**.

If you loaded sample data and you do not plan to it in the destination environment, we recommend that you delete it before you export data. **Do not delete the administrator users or administrator group**.

For information on preventing sample data from loading, refer to “Prevent Sample Data from Loading (optional)” on page 5-5.

Your database might contain samples for the following types of data:

- Product items, categories, and keywords
- Orders
- Users and groups
- Portals
- Rules
- Documents (if you use the reference implementation for content management instead of an integration with a third party)

You can use the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Administration Tool to delete sample data. To access the Administration Tool, start the server and enter the following URL in a Web browser:
`http://localhost:7501/tools/index.jsp`

For example, to delete sample users from the Administration Tool:

1. Back up your database.
2. From the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Administration Tool click User Management.
3. From the User Management page, click Users.
4. From the Users Page, under Search for Users, enter * (asterisk) in the Username box. Then click Search.

   The search mechanism returns a list of all users.
5. To delete a user, click the red X next to the username. **Do not delete the administrator users**.
6. After deleting all sample users, click Finished from the Users Page.
7. From the Users Page, click Groups.
8. From the Groups page, click the red X next to sample group names. **Do not delete the administrator group**.
9. Click Finished.

Step 5: Remove Orphaned Records

Note: In this document, `WL_COMMERCE_HOME` refers to the directory into which you installed WebLogic Commerce Server and/or WebLogic Personalization Server.

To verify that there are no orphaned records in your database, do the following:

1. In SQL*Plus, log in as SYSTEM (or another user with DBA privileges).
2. In the SQL*Plus session, enter the following command:
   ```
   @
   WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/del_orphaned
   _recs.sql
   ```
   The `del_orphaned_recs.sql` removes any orphaned records.

Step 6: Export the Data

To export data, complete **one** of the following tasks:

- If you are using the reference implementation for content management, Export All Tables in the User Account.
- If you are using a third-party tool for content management, Export Specific Tables from the User Account.

Export All Tables in the User Account

If you are using the reference implementation for content management and the Oracle user account in the destination environment contains only Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server data objects, do the following to export data:

1. From a command line, enter the following command:

   ```
   exp username/password@ORACLE_SID file=wlcs.dmp owner=WEBLOGIC
   log=wlcs_exp.log
   ```

Note: You can use a `username` and `password` for any account with DBA privileges. `ORACLE_SID` must specify the database in which the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server schema resides. If you changed the `WEBLOGIC` user account name, change the value for `owner`.

When the `exp` command finishes exporting data, it prints the following message:
`Export terminated successfully without warnings`

2. To review the results of your export, open the log file that you specified in step 1.

For more information about the Oracle export and import commands, see the following Oracle document:
http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/server.817/a76955/toc.htm.

Export Specific Tables from the User Account

Complete this procedure only if you use a third-party solution (such as Interwoven or Documentum) for content management.

If your Oracle accounts contain data objects that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server does not create and maintain (such as Interwoven and Documentum objects), do the following to export data:

1. In SQL*Plus, log in as SYSTEM (or another user with DBA privileges).
2. In the SQL*Plus session, enter the following command:

```
exp username/password@ORACLE_SID file=wlcs.dmp owner=WEBLOGIC
parfile=WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/wlcs_exp.par
log=wlcs_exp.log
```

Note: You can use a `username` and `password` for any account with DBA privileges. `ORACLE_SID` must specify the database in which the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server schema resides. If you changed the `WEBLOGIC` user account name, change the value for `owner`.

When the `exp` command finishes exporting data, it prints the following message: `Export terminated successfully without warnings`

3. To review the results of your export, open the log file that you specified in step 2.

   If you installed WebLogic Personalization Server without WebLogic Commerce Server, the log file may include `TABLE OR VIEW DOES NOT EXIST` statements for each WebLogic Commerce Server table. You can ignore these messages.

For more information about the Oracle export and import commands, see the following Oracle document:
http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/server.817/a76955/toc.htm.

Step 7: Stop the Server in the Destination Environment

Before importing data, you must stop Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server by entering `Ctrl+C` in the shell that is running the server.

Step 8: Import the Data

Caution: Before you import data, **back up your database** in the destination environment.

To import data, complete **one** of the following tasks:

- If you are using the reference implementation for content management, Import to All Tables in the User Account.
- If you are using a third party tool for content management, Import to Specific Tables in the User Account.

Import to All Tables in the User Account

If the Oracle user account in the destination environment contains only Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server data objects, do the following to import data:

1. In the destination environment, back up your database.
2. Start SQL*Plus and log in as SYSTEM (or another user with DBA privileges).

3. Enter the following command:

```
imp username/password@ORACLE_SID file=wlcs.dmp
log=wlcs_imp.log fromuser=WEBLOGIC touser=WEBLOGIC ignore=y
commit=y
```

Note: You can use a `username` and `password` for any account with DBA privileges. `ORACLE_SID` must specify the database in which the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server schema resides. If you changed the `WEBLOGIC` user account name, change the value for `fromuser` and `touser`.

For more information about the Oracle export and import commands, see the following Oracle document:
http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/server.817/a76955/toc.htm.

Import to Specific Tables in the User Account

Complete this procedure only if you use a third-party solution (such as Interwoven or Documentum) for content management.

If your Oracle accounts contain data objects that Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server does not create and maintain (such as Interwoven and Documentum objects), do the following to import data:

1. In the destination environment, back up your database.
2. Start SQL*Plus and log in as SYSTEM (or another user with DBA privileges).
3. Enter the following command:

```
imp username/password@ORACLE_SID file=wlcs.dmp log=wlcs_imp.log
parfile=WL_COMMERCE_HOME/db/oracle/8.1.6/migration/staging/wlcs_imp.par
```

Note: You can use a `username` and `password` for any account with DBA privileges. `ORACLE_SID` must specify the database in which the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server schema resides.

For more information about the Oracle export and import commands, see the following Oracle document:
http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/server.817/a76955/toc.htm.

Step 9: Start the Server in the Destination Environment

To start WebLogic Commerce Server and/or WebLogic Personalization Server on UNIX, enter the following command from a Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host:
`WL_COMMERCE_HOME/StartCommerce.sh`

To start WebLogic Commerce Server and/or WebLogic Personalization Server on Windows, on a Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server host, do one of the following:

- Click Start → Programs → WebLogic Commerce Server 3.1 → Start Commerce Server
- From a command prompt, enter the following command:
  `%WL_COMMERCE_HOME%\StartCommerce.bat`

For information on starting the server, refer to Chapter 4, “Starting and Shutting Down the Server.”

Index

Symbols

A

B

C

D

E

F

G

T

U

V

W