Installing the BEA WebLogic Commerce Servers

including the Personalization Server and Commerce Server Components

BEA WebLogic Personalization Server 2.0.1
Document Edition 2.0.1
May 12, 2000

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Installing the BEA WebLogic Commerce Servers

Document Edition	Date	Software Version
1.0	January 2000	BEA WebLogic Commerce Servers 1.7
1.1	February 2000	BEA WebLogic Commerce Servers 1.7.1
2.0	April 2000	BEA WebLogic Commerce Servers 2.0
2.0.1	May 12, 2000	BEA WebLogic Commerce Servers 2.0.1

Contents

About This Document

What You Need to Know vii
e-docs Web Site viii
How to Print the Document viii
Contact Us! ix
Documentation Conventions x

1. Preparing to Install WebLogic Commerce Servers

What's new in WebLogic Commerce Servers 1-1
Migration Issues 1-2
Overview of System Requirements 1-3
Platforms 1-3
Memory 1-3
Servers 1-4
Databases 1-4
Recommended Installation Order 1-5

2. Installing BEA WebLogic Commerce Servers

Licensing WebLogic Commerce Servers 2-1
How WebLogic Commerce Servers licensing works 2-2
How to reinstall the WebLogicCommerceLicense.xml file by hand 2-2
Installation Instructions on Windows NT 2-3
Installation Instructions on UNIX 2-5
Installation Notes 2-5
Where to Go Next 2-6

3. WebLogic Commerce Servers Configuration

Configuring Startup Files 3-2

Configuring Properties Files 3-3

Using the Cloudscape database with WLCS 3-3

Using an Oracle database with WLCS 3-4

Setting properties in the weblogic.properties file 3-4

Setting the listen port 3-4

Setting the password 3-5

Setting the document root 3-5

Deploying EJBs 3-5

Setting up connection pools 3-6

Setting up servlets and new portals 3-8

Setting up the security realm for user management 3-9

Setting properties in the weblogiccommerce.properties file 3-10

Setting up internal values 3-10

Setting up portals 3-11

Setting up the RDBMS Realm 3-12

Setting up user management tables 3-12

Setting up documentation name and location 3-12

Setting Up an Oracle Database 3-13

Recreating the Cloudscape Database 3-15

Setting Up the Reference Implementation Document Management System (DMS) 3-15

Installing and Configuring the WLCS/Interwoven TeamSite Interface 3-16

Installing and Configuring Documentum 3-20

Documentum bean environment variables 3-22

4. Example Applications

Example Applications 4-1

Portal Reference Implementation Notes 4-2

Login 4-3

User/Group Creation 4-4

Default DB Realm 4-4

Third Party Realms 4-4

Caching Properties 4-4

Repository Considerations 4-5
Setting up a portal 4-6
exampleportal Service Manager Entry 4-6

5. Migrating to WLCS 2.0.1

Portal Data Migration 5-2
Portlet definition 5-2
Portal definition 5-2
Personalization 5-2
Migrating Users 5-3
User Identification 5-3
Moving Users 5-3
Unique User Ids 5-4
Migrating Groups 5-4
Group Identification 5-4
Moving Groups 5-5
Unique Group Ids 5-5
Group-User Hierarchies 5-5
Properties and Property Sets 5-6
JSP Migration 5-6
Sample Tables 5-8
Retired Tags 5-8
New Tags 5-8
<um:login> 5-9
<um:getprofile> 5-9
<um:getproperty> 5-9
<um:setproperty> 5-9
<um:removeproperty> 5-9
<um:createuser> 5-9

About This Document

This document explains how to install the BEA WebLogic Commerce Servers including the Personalization Server and the Commerce Server Components.

This document covers the following topics:

- **Preparing to Install WebLogic Commerce Servers**: Pre-installation information to understand when installing the system.
- **Installing BEA WebLogic Commerce Servers**: The installation procedure for WebLogic Commerce Servers on NT and supported UNIX systems.
- **WebLogic Commerce Servers Configuration**: System configuration information.
- **Example Applications**: Sample portals that ship with WebLogic Commerce Servers 2.0.1.
- **Migrating to WLCS 2.0.1**: Migration information for moving from WebLogic Commerce Servers 1.7 to 2.0.1.

What You Need to Know

This document is intended for business analysts, Web developers, and Web site administrators involved in setting up an eCommerce site using BEA WebLogic Commerce Servers. It assumes a familiarity with the WebLogic Commerce Servers platform and related Web technologies as described below. The topics in this document are organized primarily around development goals and the tasks needed to accomplish them. Generally, a set of topics also speaks to a particular development role and requires the basic knowledge with regard to the technology focus of that role:

- *HTML author* uses the Java Server Page (JSP) tags provide in the JSP tag library, thereby leveraging the power of personalization without having to know Java.
- *Java Server Page (JSP) developer* creates JSPs using the tags provided or by creating custom tags as needed.
- *Application assembler*, *system analyst*, or *systems integrator* writes rules, writes, schemas, and monitors usage.
- *System administrator* installs, configures, deploys, and monitors the Web application server
- *Java developer* extend or modifies the Enterprise Java Bean (EJB) components that make up the Commerce Server engine, if that level of customization is needed.

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://edocs.bea.com.

How to Print the Document

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

A PDF version of this document is available on the WebLogic Commerce Servers 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 WebLogic Commerce Servers 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/.

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Commerce Servers 2.0.1 release.

If you have any questions about this version of BEA WebLogic Commerce Servers, or if you have problems installing and running BEA WebLogic Commerce Servers, contact BEA Customer Support through BEA WebSupport at **www.beasys.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 file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR

Convention	Item
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[ ]	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: buildobjclient [-v] [-o name] [-f *file-list*]... [-l *file-list*]...
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: buildobjclient [-v] [-o name] [-f *file-list*]... [-l *file-list*]...
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

1 Preparing to Install WebLogic Commerce Servers

Preparing to Install WebLogic Commerce Servers includes the following topics:

What’s new in WebLogic Commerce Servers

Migration Issues

Overview of System Requirements
- Platforms
- Memory
- Servers
- Databases

Recommended Installation Order

What’s new in WebLogic Commerce Servers

- **Personalization Advisor**: Personalization Advisor recommends content and performs several important functions in creating a personalized application, including searching for content, tying the other core personalization services together, and matching content to user profiles.

- **Property Set Management**: Property Set Management allows you to create property sets, the schema of personalization attributes, and the properties that make up property sets.
- **User Management**: User Management joins enterprise data about users with profile data that is used to personalize the users' view of the application.
- **Content Management**: The Content Management component provides content and document management capabilities for use in personalization services. You can use the document management system (DMS) provided as a reference implementation with WLPS 2.0.1, or you can use a third-party vendor's DMS, including Interwoven's TeamSite/OpenDeploy product and Documentum's 4i product.
- **Rules Management**: The Rules Management component allows developers to create business rules that turn on and off content and match content to users according to user profile information.

Migration Issues

The WLCS 2.0.1 installation will not affect data in existing Portal 1.7 implementations. Refer to "Migrating to WLCS 2.0.1" on page 5-1 for more information about migration issues related to installing WLCS 2.0.1 with an existing Portal 1.7 installation.

Overview of System Requirements

This section presents general information about the WebLogic Commerce Servers system requirements. **For the latest platform updates and version-specific details**, see the Supported Platforms section of the WebLogic Commerce Servers Release Notes. The Release Notes document resides on the BEA "e-docs" Web site and is updated periodically. For example, it is updated when the WebLogic Commerce Servers software supports an additional platform vendor and operating system. For our hardcopy readers, the URL of the Supported Platforms section is:

`http://edocs.bea.com/wlcs/docs20/relnotes/relnotes.htm#platforms`

Platforms

BEA WebLogic Commerce Servers (WLCS) is available for Microsoft Windows NT systems and various UNIX systems. For the latest details, see the Supported Platforms section of the WLCS Release Notes.

Note: For Windows NT, it is strongly recommended that you be logged into the PC as a user with administrative privileges prior to installation.

Memory

The product requires at least 128 MB of memory (RAM) to install and run. The system will run much better with more RAM, though. The installation requires the following free disk space:

- 50 MB (Windows)
- 40 MB (UNIX)

Servers

WLCS requires the following software products:

- WebLogic Server 5.1 (SP1 for WLCS Components)
- Java 2 SDK 1.2.

  Note: Different Java 2 SDK 1.2.* version levels are required for each platform. For the latest details, see the Supported Platforms section of the WLCS Release Notes. You must use at least JDK 1.2 to run the example applications that ship with the product.

- Web browser (Internet Explorer 4.0 and above, or Netscape Communicator 4.5 and above)
- Text editor (such as Notepad, emacs, VI)
- Rational Rose 98i or 2000 required for use of the Rose Add-In, Rose UML, and SmartGenerator (for the WLCS Components)

For the latest details, see the Supported Platforms section of the WLCS Release Notes.

Databases

- A DBMS and a JDBC driver for the DBMS. This release has been certified for Cloudscape and Oracle databases. For the latest details about the supported database version numbers and drivers, see the Supported Platforms section of the WLCS Release Notes.
- An evaluation version of Cloudscape is distributed with the product as part of the example implementations.

Recommended Installation Order

For a new installation, BEA recommends you install the components in the following order:

1. JDK
2. Rational Rose
3. WebLogic Server
4. WebLogic Commerce Servers (WLCS)

2 Installing BEA WebLogic Commerce Servers

Installing BEA WebLogic Commerce Servers includes the following topics:

Licensing WebLogic Commerce Servers

- How WebLogic Commerce Servers licensing works
- How to reinstall the WebLogicCommerceLicense.xml file by hand

Installation Instructions on Windows NT

Installation Instructions on UNIX

Installation Notes

Where to Go Next

Licensing WebLogic Commerce Servers

When you register to download WebLogic Commerce Servers (WLCS), you should also download a current evaluation license. If you received the product on a disk, you will receive a license file via e-mail to the e-mail account specified on the purchase order.

Note: If you need a new evaluation license, revisit commerce.beasys.com, go through the download process, and download only the evaluation license.

How WebLogic Commerce Servers licensing works

WebLogic Commerce Servers license keys are stored in `WL_COMMERCE_HOME/license` in a file called `WebLogicCommerceLicense.xml`. This file must reside either in your working directory or in a directory included in your `WEBLOGIC_CLASSPATH`.

Note: WLCS ships with the license file in the `CLASSPATH`. If you have problems starting the server, we recommend you check the `WL_COMMERCE_HOME\license` directory and add the directory to your `WEBLOGIC_CLASSPATH` if necessary. Moving the `WebLogicCommerceLicense.xml` file to a directory that is already in your `CLASSPATH` will also work.

How to reinstall the WebLogicCommerceLicense.xml file by hand

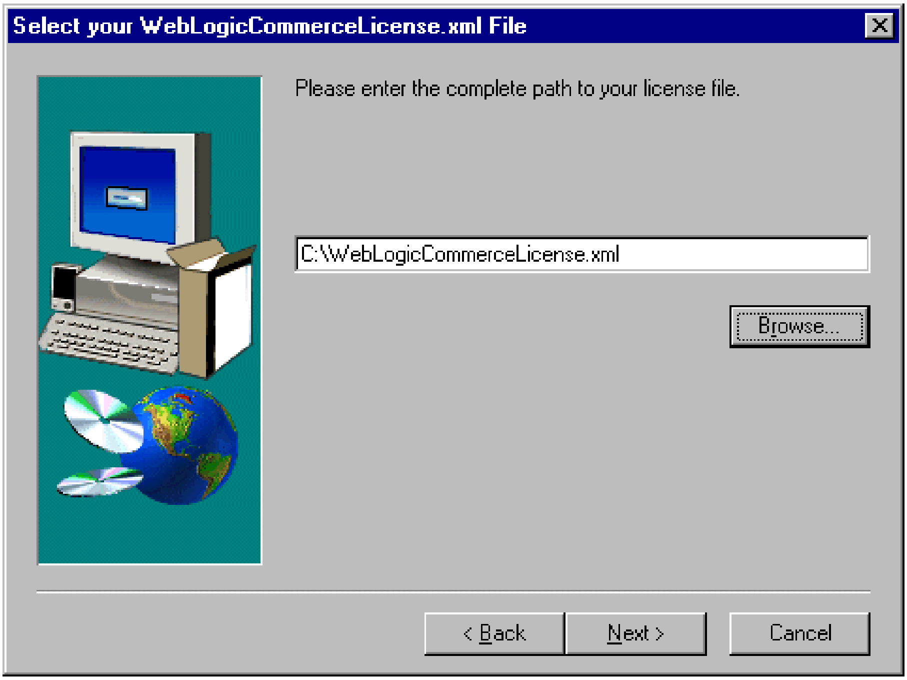
If you receive a new `WebLogicCommerceLicense.xml` and need to reinstall it, you should:

1. Delete any old versions of `WebLogicCommerceLicense.xml` from your `WEBLOGIC_CLASSPATH`.
2. Verify that the `WebLogicCommerceLicense.xml` file is in your `CLASSPATH` or working directory. The easiest approach is to keep your `WebLogicCommerceLicense.xml` in the `WL_COMMERCE_HOME/license` directory and ensure this directory is in your `WEBLOGIC_CLASSPATH`.

Note: If you have two license statements for the same product in the file, only the first one will be used.

Installation Instructions on Windows NT

1. Start the installation by opening the installation package, `WLCS_201_NT4.exe`.
2. Click through the Welcome and Software License Agreement screens.
3. In the Select your WebLogicCommerceLicense.xml screen, point the wizard to the WebLogic Commerce Server license (`WebLogicCommerceLicense.xml`) on your server. Click **Next** after you locate the license file.


Select your WebLogicCommerceLicense.xml File

Please enter the complete path to your license file.

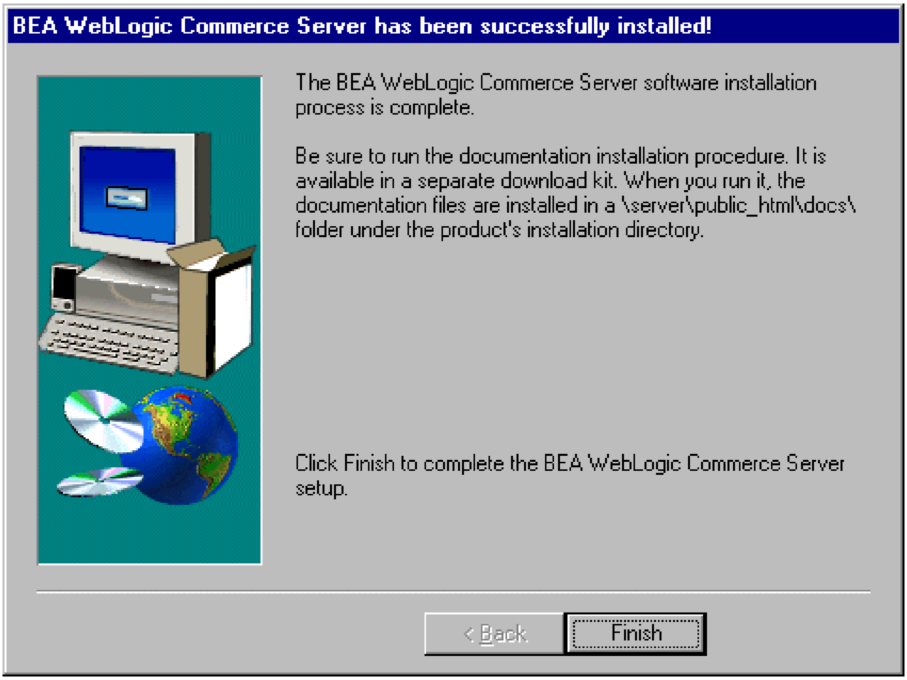
C:\WebLogicCommerceLicense.xml

4. The InstallShield autodetects your JDK, WebLogic Server, and Rational Rose installations. If it doesn't find one of those items, it asks you to point to the directory that contains it.

   Note: Rational Rose is required for certain features in the Components, but it's not required to run the system. Refer to the Components documentation for information about using Rational Rose with the Components.

5. In the Select Destination Directory screen, point to the installation directory you want to use. The default installation directory is `\weblogicCommerce`. Click **Next**.

6. After the installation completes, the BEA WebLogic Commerce Server has been successfully installed screen appears.



BEA WebLogic Commerce Server has been successfully installed!

The BEA WebLogic Commerce Server software installation process is complete.

Be sure to run the documentation installation procedure. It is available in a separate download kit. When you run it, the documentation files are installed in a \server\public_html\docs\ folder under the product's installation directory.

Click Finish to complete the BEA WebLogic Commerce Server setup.

7. Run `StartCommerce.bat` to start the Cloudscape version of WebLogic Commerce Servers 2.0.1.

   Note: See “Setting Up an Oracle Database” on page 3-13 to run WebLogic Commerce Servers with a database other than the Cloudscape database that ships with the system.

Installation Instructions on UNIX

1. Unjar the WLCS_201_UNIX.zip installation file in a directory. This process creates a directory called /WebLogicCommerce.
2. Change to the /WebLogicCommerce directory and run install.sh to create the startup files for WebLogic Commerce Servers. You need to know the location of the JDK and WebLogic Server to complete the installation.
3. Make a license directory at <install-dir>/license.
4. Copy your WebLogicCommerceLicense.xml file to the new <install-dir>/license directory.
5. Run StartCommerce.sh to start the Cloudscape version of WebLogic Commerce Servers 2.0.1.

   Note: You may need to change file permissions to run the script.

Installation Notes

WLCS 2.0.1 ships with two supported database configurations: Oracle and Cloudscape. To enable easier deployment of WLCS with these databases, BEA provides configuration files for each database configuration. The *-cmp files specify Cloudscape configuration and *-bmp files specify Oracle configuration. You may need to copy these files into the default weblogic.properties and weblogiccommerce.properties files, depending on your database. Refer to "Configuring Properties Files" on page 3-3 for more information about configuring the files.

Where to Go Next

1. Change to the installation directory and start WLCS by running `StartCommerce.bat` (`StartCommerce.sh` for UNIX). The default installation path is `\weblogicCommerce\StartCommerce.bat` or `/weblogicCommerce/StartCommerce.sh`. You will see the console output in the console window while the server initializes. You will know when the server has finished starting when you see the message `WebLogic Server started` in the console output.

   Note: This file defaults to the Cloudscape database. See "Setting Up an Oracle Database" on page 3-13 for information about using Oracle databases.

2. Using a browser, try the example applications to make sure the server is working correctly. The example applications use the Cloudscape database and reference implementation DMS that ship with the system; therefore, you should not have to make any configuration changes to test the system. All you need to do is point your browser to the correct URL listed for each application below.

   Note: Refer to "WebLogic Commerce Servers Configuration" on page 3-1 for information about making configuration changes to the system. You need to change configurations if you use a database other than Cloudscape or a DMS other than the reference implementation DMS shipped with WLCS 2.0.1. The configuration document also contains information about configuring other pieces of the system, including servlets and portals.

   You might want to leave the console window in view when testing the example applications. You can see the console output indicating that the server is hitting JSP pages, compiling the Java, and taking care of other chores.

   Note: Each page loads slowly the first time you access it because the Java compiler has to compile the page source code. Subsequent loads and refreshes of the page go much faster than the initial load.

- The MyBuyBeans sample application is included as an example of the WebLogic Commerce Servers power. The application demonstrates the commerce-based components of the system and the personalization services. You can view the source in `public_html/buybeans` and `public_html/portals/buybeans`.

To understand how directory structures relate to the URLs, refer to “Setting up servlets and new portals” on page 3-8. To run the application, point your browser to `http://<wl-host>:<port>/mpbb`.

Note: `/mpbb` is an entry page for the MyBuyBeans portal that requires you to log in with an existing user. You can hit `http://<wl-host>:<port>/mybuybeans` to go directly to the MyBuyBeans portal without logging in and then create a new user with the tools provided in the upper-right corner of the page.

Note: `<port>` is the `ListenPort` of the running WebLogic Server set in the `weblogic.properties` file. The default value for `ListenPort` is `7601`.

- View the Acme Portal example at `http://<wl-host>:<port>/exampleportal`. This portal demonstrates functionality of the Portal Management and User Management components.

- For personalization administration needs, point your browser to `http://<wl-host>:<port>/wlpsadmin` for the WebLogic Personalization Server administration tools. These tools allow you to create and manage property sets, users, rules, and portals.

3. Go through the tutorial that ships with the documentation. The tutorial extends the MyBuyBeans.com portal to include personalization services for two different user groups.

4. Begin building and deploying your own personalized applications using servlets or portals. Refer to the documentation for more information about using the system.

3 WebLogic Commerce Servers Configuration

WebLogic Commerce Servers Configuration includes the following topics:

Configuring Startup Files

Configuring Properties Files
- Using the Cloudscape database with WLCS
- Using an Oracle database with WLCS
- Setting properties in the weblogic.properties file
- Setting properties in the weblogiccommerce.properties file

Setting Up an Oracle Database

Recreating the Cloudscape Database

Setting Up the Reference Implementation Document Management System (DMS)

Installing and Configuring the WLCS/Interwoven TeamSite Interface

Installing and Configuring Documentum
- Documentum bean environment variables

After installing WebLogic Commerce Servers, you might need to configure files, setup a database, set up the Document Management System (DMS), or perform other tweaks to the default installation. This document describes the changes to make to configure the server.

Note: Most default settings will work fine out of the box, especially if you run the server with the Cloudscape database and reference implementation DMS that ships with WLCS. If you use another database or DMS or if you need configuration information for setting up connection pools, servlets, security, etc., this document includes configuration information for these tasks.

Configuring Startup Files

WLCS uses two files to set environment variables when it starts the server: `sethome.bat` (`sethome.sh` for UNIX) and `StartCommerce.bat` (`StartCommerce.sh` for UNIX). Use `StartCommerce` to start WLCS.

- `StartCommerce.bat` (`StartCommerce.sh` for UNIX)
  - This file starts the server. It calls `sethome.bat` or `sethome.sh` to configure the environment settings and then starts the server. You don’t need to make changes to the `StartCommerce.bat` or `StartCommerce.sh` file.
- `sethome.bat` (NT in `<install-dir>\bin\win32`) and `sethome.sh` (UNIX systems in `<install-dir>/bin/solaris2`)

  Note: This file contains most of the environment settings and is called by the `StartCommerce` script. The installation program configures the settings during installation. You should not have to change it if you use the Cloudscape database that ships with WLCS 2.0.1.

  - If you have problems with the server starting, ensure `WL_COMMERCE_HOME` is set to the directory that contains the WLCS install. Also ensure the `DB_CLASSPATH` property points to your database drivers and the `JDK_HOME` points to your JDK. All other settings should work as shipped.

Configuring Properties Files

This section contains detailed reference information for configuring the properties files for WLCS. The topics here discuss setting up connection pools, servlets, and portals, deploying EJBs, and setting up security realms, in addition to other WLCS properties.

Note: To run the Cloudscape database with the reference DMS that ships with the system, you should not have to configure any of the properties files here. The configuration options in this document cover database setup information for Oracle, DMS setup for Interwoven and Documentum products, and a variety of smaller topics relating to Service Manager entries, realms, security, etc.

The `weblogic.properties` and `weblogiccommerce.properties` files that ship with the system have corollary files with configuration information for the Cloudscape and Oracle databases. These corollary files, `weblogic-cmp.properties` and `weblogiccommerce-cmp.properties` for Cloudscape and `weblogic-bmp.properties` and `weblogiccommerce-bmp.properties` for Oracle, contain the default configuration information for each database. These *stock* files serve as templates for your `weblogic.properties` and `weblogiccommerce.properties` files.

Note: The `weblogic.properties` and `weblogiccommerce.properties` files that ship with the system contain the default Cloudscape database configuration information that is also contained in the corollary files, `weblogic-cmp.properties` and `weblogiccommerce-cmp.properties` files. If you use the Cloudscape database, you don’t need to edit the `weblogic.properties` and `weblogiccommerce.properties` files.

Using the Cloudscape database with WLCS

Out of the box, the `weblogic.properties` and `weblogiccommerce.properties` files contain the same information as the `weblogic-cmp.properties` and `weblogiccommerce-cmp.properties` since the Cloudscape database runs the example applications.

Note: If you plan to use the Cloudscape database, you should not need to make modifications to the `weblogic.properties` and `weblogiccommerce.properties` files.

Using an Oracle database with WLCS

If you use an Oracle database, edit the following properties files, `weblogic-bmp.properties` and `weblogiccommerce-bmp.properties`, then copy the files over the `weblogic.properties` and `weblogiccommerce.properties` files.

Note: Basically, you need to rename `weblogic-bmp.properties` and `weblogiccommerce-bmp.properties` to `weblogic.properties` and `weblogiccommerce.properties` and make sure they're in the directory specified in the `StartCommerce` file (by default, the installation directory). This will enable WLCS to use an Oracle database instead of the default Cloudscape database. See "Setting Up an Oracle Database" on page 3-13 for more information about configuring the databases.

You also need to change `DEPLOYMENT_SET=cmp` to `DEPLOYMENT_SET=bmp` in `sethome.bat`.

Setting properties in the weblogic.properties file

The following values need to be set in the `weblogic.properties` file for the WebLogic Server instance to use as the host for the WLCS components. A `weblogic.properties` file is provided in the installation.

Setting the listen port

Look in the entry for the TCP/IP port number in the `weblogic.properties` file in the Core System Properties section. You can change this value, and the default is `7601`. The default entry looks like the following snippet:

```
# TCP/IP port number at which the WebLogic Server listens
    for connections
weblogic.system.listenPort=7601
```

Setting the password

In the Core Security-Related Properties of `weblogic.properties`, the following entry sets the username and password. Note that `system` is the username and `weblogic` is the password for the entry. You need to enter these values when using the WLPS Administration Tools at `http://<wlhost>:<port>/wlpsadmin`.

```
weblogic.password.system=weblogic
```

Setting the document root

The document root must be set to include pages for the administration tools, portal pages, and other pages to be accessible via the web server. For example, if you create create a new portal, then its entries for `homepage`, `workingdir`, and `defaultdest` will be relative to the document root.

```
weblogic.httpd.documentRoot=/weblogicCommerce/server/public_html
```

Note: The document root should be set correctly out of the box. This is the default location and should not change unless you move the directory.

Deploying EJBs

The following jar files must be included in the `weblogic.ejb.deploy` property value to properly deploy the WLCS beans.

Note: There is no need for a drive specification if you're deploying jar files from the same file system of the `<install-dir>`. However, if you want to deploy jar files from another file system, then you should specify the complete path.

Example (installation directory file system):

Note: The first four jar files are different for Cloudscape and Oracle entries. Use the `*-cmp` jar files for Cloudscape deployments.

```
weblogic.ejb.deploy=\
        /weblogicCommerce/lib/axiom-bmp-deploy.jar,\
        /weblogicCommerce/lib/ebusiness-bmp-deploy.jar,\
        /weblogicCommerce/lib/foundation-bmp-deploy.jar,\
        /weblogicCommerce/lib/examples-bmp-deploy.jar,\
        /weblogicCommerce/lib/document.jar,\
        /weblogicCommerce/lib/usermgmt.jar,\
        /weblogicCommerce/lib/bridge.jar,\
        /weblogicCommerce/lib/foundation.jar,\
        /weblogicCommerce/lib/axiom.jar,\
        /weblogicCommerce/lib/portal.jar,\
        /weblogicCommerce/lib/ruleeditorbeans.jar,\
        /weblogicCommerce/lib/rulesservice.jar,\
        /weblogicCommerce/lib/schemamgr.jar,\
        /weblogicCommerce/lib/p13nadvisor.jar
```

Setting up connection pools

A JDBC jts pool, a typical JDBC connection pool, and a special connection pool to support the content management component must all be created to support the WLCS components. The following examples provide a guideline for setting up these pools. A developer needs to adjust the pool properties to properties appropriate for the deployment configuration.

Note: Use `weblogic-cmp.properties` for Cloudscape and `weblogic-bmp.properties` for Oracle and copy the file over `weblogic.properties`.

Example (using Cloudscape):

```
weblogic.jdbc.DataSource.weblogic.jdbc.jts.commercePool=
    commercePool

weblogic.jdbc.connectionPool.commercePool=\
       url=jdbc:cloudscape:Commerce,\
       driver=COM.cloudscape.core.JDBCDriver,\
       initialCapacity=5,\
       maxCapacity=10,\
       capacityIncrement=1,\
       props=user=test;password=test

# Add an ACL for the connection pool:
weblogic.allow.reserve.weblogic.jdbc.connectionPool.commercePool=
everyone
```

```
# this is the pool the beans reference
weblogic.jdbc.connectionPool.docPool=\
url=jdbc:beasys:docmgmt:com.beasys.commerce.axiom.document.ref.Re
fDocumentProvider,\
       driver=com.beasys.commerce.axiom.document.jdbc.Driver,\
       loginDelaySecs=1,\
       initialCapacity=1,\
       maxCapacity=5,\
       capacityIncrement=1,\
       allowShrinking=true,\
       shrinkPeriodMins=15,\
       refreshMinutes=10,\
       props=jdbc.url=jdbc:weblogic:pool:commercePool;\
             jdbc.isPooled=true;\
             docBase=/weblogicCommerce/dmsBase/;\
             schemaXML=/weblogicCommerce/dmsBase/doc-schema.xml
weblogic.allow.reserve.weblogic.jdbc.connectionPool.docPool=every
one
```

Example (using Oracle):

```
# Add a DataSource entry for JDBC connections. You will then
# reference it as:
# Initial Context initContext = new InitialContext();
# DataSource ds =
#    (javax.sql.DataSource)initContext.lookup(“weblogic.jdbc.jts.
#    commercePool”);
weblogic.jdbc.DataSource.weblogic.jdbc.jts.commercePool=commerceP
ool

# In the entry below, server is hostname of database and
#    TNSNAMES is the entry in tnsnames.ora
# Example: url=jdbc:oracle:thin@moab:1521:TEKA
weblogic.jdbc.connectionPool.commercePool=\
       url=jdbc:oracle:thin:@server:port:TNSNAME,\
       driver=oracle.jdbc.driver.OracleDriver,\
       initialCapacity=5,\
       maxCapacity=10,\
       capacityIncrement=1,\
       props=user=test;password=test

# Add an ACL for the connection pool:
weblogic.allow.reserve.weblogic.jdbc.connectionPool.commercePool=
everyone
```

```
# Add an ACL for the connection pool:
weblogic.allow.reserve.weblogic.jdbc.connectionPool.jdbcPool=ever
yone
```

Note: If you use the `com.beasys.commerce.axiom.contact.security.RDBMSRealm` class, settings in the `weblogiccommerce.properties` file for the security realm must match these database settings.

Setting up servlets and new portals

Note: Leave the default entries for the demo portals and servlets that ship with WLCS 2.0.1. You only need to add entries for new portals and servlets.

The `PortalServiceManager` registration for `exampleportal` below shows an example of how to register a portal; the specific parameters are described in the section on Creating and Managing Portals.

Note: You should include the `wlpsadmin` tools entries directly.

You can view a table listing the valid `PortalServiceManager` parameters and values in the Creating and Managing Portals section. In the entry below, note the following name/value pairs:

- `weblogic.httpd.register.exampleportal`: URL of the portal or servlet. For example, `http://<wlhost>:<port>/exampleportal`

  Note: The default `ListenPort` value is `7601`.

- `portalname=exampleportal`: Name of the portal assigned using the Portal Management administration tool.
- `homepage, defaultdest,` and `workingdir` locations are relative to `weblogic.httpd.documentRoot`

Examples:

```
weblogic.httpd.register.exampleportal=com.beasys.commerce.portal.
admin.PortalServiceManager
weblogic.httpd.initArgs.exampleportal=\
    portalname=exampleportal,\
    homepage=/portals/example/portal.jsp,\
    defaultdest=/portals/example/portal.jsp,\
    workingdir=/portals/example/,\
    groupname=AcmeUsers,\
sessioncomparator=com.beasys.commerce.portal.admin.PortalSessionC
omparator,\
    refreshworkingdir=120,\
    repositorydir=/portals/repository/,\
    timeout=99999,\
    allowautologin=false

weblogic.allow.execute.weblogic.servlet.wlpsadmin=system
```

Setting up the security realm for user management

The weblogic.security.realmClass must be set to the appropriate WLCS class to handle authentication and to provide user and group information. It is unlikely a developer needs to change this value unless using external sources such as LDAP.

Example (in weblogiccommerce.properties):

```
weblogic.security.realmClass=com.beasys.commerce.axiom.contact.
security.RDBMSRealm
```

Note: This is for the default realm only.

The RDBMSRealm properties must be set according to the database setup for the installation. A developer is likely to need to adjust these values based on the deployment configuration. The example below shows entries for both Cloudscape and Oracle. Note that the dbUrl must match that of the JDBC connection pools as defined in the weblogiccommerce.properties file.

Example (in weblogiccommerce.properties):

```
#-------------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

#---------------Oracle-------------------#
commerce.usermgmt.RDBMSRealm.driver=oracle.jdbc.driver.OracleDriver
commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:oracle:thin:@server:port:TNSNAME
commerce.usermgmt.RDBMSRealm.dbUser=test
commerce.usermgmt.RDBMSRealm.dbPassword=test
```

Note: This is for the default realm only.

Setting properties in the weblogiccommerce.properties file

The `weblogiccommerce.properties` file is used to set values for the WLCS components. A sample `weblogiccommerce.properties` file is included in the installation.

Setting up internal values

These properties are used to set up values to be used internally to identify pools, classes, and home names. It is unlikely a developer would ever need to change these values, so these lines should remain unchanged.

Example:

```
commerce.jdbc.pool.name=jdbcPool
commerce.jts.pool.name=commercePool

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.home.p13n.AdvisorHome=com.beasys.commerce.axiom.p13n.advisor.Personali
zationAdvisorHome
commerce.home.rules.RulesServiceHome=com.beasys.commerce.axiom.reasoning.rules.
service.RulesServiceHome
commerce.home.property.SchemaManagerHome=com.beasys.commerce.foundation.propert
y.SchemaManager
```

Setting up portals

The portal component has a number of properties that it uses as defaults for when a new portal gets created. A developer might want to adjust these values based on the nature of the portal being built. The defaults can be overridden using the portal administration tools.

Note: These portals will be copied from the `weblogic.httpd.documentRoot/<repository>` directory into the corresponding `portal/` directories if they do not exist already.

Example:

```
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.portal.password=guest
```

The following property tells the portal component which property set to use as the default for portal users. A developer may want to change this value based on property sets created using the Property Set Management administration tool.

Example:

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

The following portal properties are for internal use and should not be altered by a developer.

Example:

```
commerce.application.minimum.build=57998
commerce.default.category.name=Home
```

Setting up the RDBMS Realm

If you need to change the settings in the weblogiccommerce.properties file for the RDBMSRealm, change the database settings in a similar fashion to the database pools in weblogic.properties. Set the dbDriver, dbUrl, dbUser, and dbPassword to use the same database that the commercePool uses in weblogic.properties.

Setting up user management tables

The following properties are used to identify the internal names for tables for use by the user management component. It is unlikely a developer would ever need to change these values.

Example:

```
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
```

Setting up documentation name and location

The following properties are used to set up the documentation for the Personalization Server components.

Example:

```
commerce.p13n.document.dev.name=index.html
commerce.p13n.document.dev.version=20
commerce.p13n.document.dev.docRootDir=docs
```

Setting Up an Oracle Database

For information on the Oracle database and driver that this release supports, see the Supported Platforms section of the WLCS Release Notes.

For development and production purposes, WLCS requires a DBMS with specific database configurations. This WLCS distribution includes several SQL scripts that create the initial database information.

Note: You must have a JDBC driver and have configured connection pools (EJB) to access your database in the WebLogic Server.

To create a new Personalization DBMS for Oracle databases:

1. Create a new database user.
2. Make sure the user has all the necessary permissions for creating and modifying tables, indices, etc.
3. Log in to an SQL tool, such as SQL Worksheet, and run one of the following scripts, depending on the client's operating system. The scripts are located in `<install-dir>\db\oracle.` For NT clients, use `create-p13n-oracle-nt.sql`. For UNIX clients, use `create-p13n-oracle-unix.sql`. Note that if you use a remote database, the database host machine's operating system does not affect the script; you should run the script that matches the client machine.

   The SQL scripts create the database schemas, including both the commerce tables and personalization tables, and populate the tables with default data. However, the example applications (see "Example Applications" on page 4-1 for more information) will not work without completing Step 4 below to populate the remainder of the database.

Note: If the scripts are not in the default directory, change the path in `<install-dir>\db\oracle\createp13n-oracle.sql` to match the correct path. Then just run this file alone and reference the others.

4. Update the `DB_CLASSPATH` environment variable within the `sethome.bat` (`sethome.sh` for UNIX) in the installation `bin\win32` (`bin/solaris2` for UNIX systems) directory to include your JDBC database drivers.

   Note: The `DB_CLASSPATH` is preset with hooks into the Cloudscape database. To use an Oracle database, change this environment variable to your DBMS JDBC database driver in `sethome.bat` or `sethome.sh`. The entry should be one of the following two entries:

   - Cloudscape: `SET DEPLOYMENT_SET=cmp`
   - Oracle: `SET DEPLOYMENT_SET=bmp`

5. Configure your database connection pools.

   a. Open the `weblogic-bmp.properties` file that resides in your installation directory.

   b. Change the `commercePool` connection properties to reflect your database configuration.

   Note: The properties file is preset with hooks into the Cloudscape database. So change the `commercePool` entries to reflect your database. For more information on the JDBC pools, see http://www.weblogic.com/docs/admindocs/properties.html#conpools. Make sure you edit the appropriate `*-cmp` (Cloudscape) or `*-bmp` (Oracle) database configuration files. Make sure you copy the Oracle file over the `weblogic.properties` file.

   c. Make the same change in `weblogiccommerce-bmp.properties` and then copy it over `weblogiccommerce.properties`.

6. Populate the remainder of the tables by performing the following steps:

   a. Start WebLogic Server.

   b. Run `DataLoader.bat` (`<install-dir>\bin\win32` for NT) or `DataLoader.sh` (`<install-dir>/bin/solaris2` for UNIX systems) from the appropriate directory to populate the tables.

   c. Restart WebLogic Server.

Recreating the Cloudscape Database

You might need to recreate the default Cloudscape database if your Cloudscape database becomes corrupted or to clear out the current tables. To recreate the database:

1. Run `BuyBeansDropCloudscape.sql` to remove the buybeans tables from the database in the existing Cloudscape database.
2. To create the new database, run `CloudscapeDatabaseCreator.bat` (NT) or `CloudscapeDatabaseCreator.sh` (UNIX) in `<install-dir>\db\cloudscape`.

Setting Up the Reference Implementation Document Management System (DMS)

This release ships with a default DMS using the WLPS reference implementation. To index content, the DMS uses the BulkLoader, a command-line utility that indexes all content under the `\dmsBase` directory. Use the `loaddocs.bat` (NT) and `loaddocs.sh` (UNIX) files to run the BulkLoader. Refer to the Content Management component documentation for detailed information about the command-line switches provided for the BulkLoader utility.

See “Installing and Configuring the WLCS/Interwoven TeamSite Interface” on page 3-16 for information about using Interwoven’s TeamSite DMS and see “Installing and Configuring Documentum” on page 3-20 for information about using the Documentum DMS.

Note: The demo database ships with the default content already indexed. For UNIX systems, change `sethome.sh` and `loaddoc.sh` files in `<install-dir>/bin/solaris2/` to contain the paths to the installation directory.

Installing and Configuring the WLCS/Interwoven TeamSite Interface

All configuration documented in this section applies to the Interwoven TeamSite Server. If you are not using Interwoven TeamSite for content management, skip this configuration procedure.

Installing and configuring the WLCS interface begins only after successful installation of version 4.2 of the TeamSite, TeamSite Templating, Open Deploy and Data Deploy components, and version 5.1 of the WebLogic server and version 2.0 of the WebLogic Commerce Server. This includes configuration and testing of the Data Deploy Automation Server (DAS) component.

The installation of the WLCS interface involves the following steps:

1. Install the WLCS database server client interface on the TeamSite server.

The current version of the WLCS interface executes on the TeamSite server and may require certain components to be installed to provide remote connection with the WCLS database. This step is unnecessary if the WLCS database and TeamSite server are on the same platform. In addition, there may be environment variables that must be defined to allow a valid connection. Refer to vendor documentation for specific installation requirements.

2. Modify the database connection parameters to access the WLCS database.

The DNR scripts, *teamsite_wlcs_trans.ipl* and *teamsite_wlcs.ipl*, require a user id and password for an account with access to the WLCS database and a valid Data Source Name (DSN) as defined by the Perl Data Base Interface (DBI) standard. The scripts are written in Perl and use the DBI interface for accessing all databases – both TeamSite and WLCS. The connection command to the destination database must be modified to use these parameters to establish a connection. This command is identified in each of the scripts as shown below:

# The following is used to connect to the database. The format is:

```
# DataSourceName, User, Password
# Replace the hard-coded User ID and Password (scott, tiger) with
a valid account
# for the destination database. Replace the Data Source Name with
a valid
# specification for the destination database. The DSN is database
specific.
```

```
my $DES = DBI->connect("DBI:ODBC:ORAIWOV","scott","tiger",
{RaiseError => 1} );
```

In the above example, the parameter *scott* would be replaced with a valid Oracle User Id, *tiger* would be replaced with the corresponding password, and ORAIWOV would be replaced with a valid Data Source Name (DSN). The DSN is the most likely source of error. On NT resident databases, the easiest approach to defining a valid DSN is to create a system DSN via the ODBC control panel component. For a Unix database target the DSN is normally the name of the database server host, its listening port (for example, Oracle uses 1521 as the default port number), and sometimes the database name. For example, an Oracle database DSN could be specified:

```
DBI:ODBC:<server host name>:<port>
```

If the target database platform provides a DBD driver that can be installed on the TeamSite server (client) then that driver can be used instead of the ODBC DBD driver shown above. For example:

```
DBI:ORACLE:<server host name>:<port>
```

Refer to the vendor specific documentation for the correct format of the DSN specification. The parameters used can be tested using the *dbish* utility provided with TeamSite 4.2.

3. Modify the TeamSite Data Deploy database connection parameters.

This step is optional depending on the TeamSite Data Deploy configuration chosen when setting up the Data Deploy Automation Server (DAS). The current implementation does not attempt to access those configuration files to determine what options were selected. The default configuration uses the SQL Anywhere database that ships with TeamSite. This is also the configuration that the DNR scripts use by default. Therefore, if all defaults were taken, then the DNR script should require no modification. Please note, however, that the SQL Anywhere license provided with TeamSite only supports a single connection. This means that multiple deployments cannot be run in parallel, and, that DAS actions cannot occur simultaneously with deployment. A warning message is generated and the deployment terminated if this condition occurs.

4. Modify the location of the mime.types file.

The interface looks up the mime type of a document based on its extension using the standard mime.types file. The location of this file needs to be specified in the DNR script. This is done by setting the value of the $MIME_TYPE_PATH variable as shown in the following example (Note: make sure the path ends in either a "\", for NT clients, or a "/" for Unix clients):

```
$MIME_TYPE_PATH = "D:\"
```

Note: If not specified, the mime.type file must reside with the DNR scripts.

5. Modify the Open Deploy client configuration file.

This step is a standard part of using Open Deploy's DNR feature. The client configuration file must be edited to specify the path to the DNR script (either *teamsite_wlcs_trans.ipl or teamsite_wlcs.ipl* should be selected depending on whether transaction mode is required) and a single parameter that identifies the EDITION being deployed.

Note: The illustration below inserts newlines for readability. These should not be left in the client configuration file; refer to the Administering Open Deploy manual.

```
deployment=webEdition
    teamsite_based
    area=//IWSERVER/default/main/www/EDITION
    previous_area=//IWSERVER/default/main/www/EDITION/IW_PREV
    deploy_run_script=d:\iw-home\iw-perl\bin\iwperl.exe
       d:\teamsite_wlcs_trans.ipl
      //IWSERVER/default/main/www/EDITION
      when=client_after_deploy
.
.
.
;
```

- This is a transaction mode, teamsite-based deployment. This is the recommended configuration. A directory based deployment from a staging area is supported but it will deploy all data from the staging area Data Deploy based tables, whether new or modified, and will not remove data records that were previously deployed and deleted from the staging area.

- The area (area and previous area) must either be an edition or a staging area. The string *EDITION* in the area version path (vpath) is used to identify an edition deployment area. The string *STAGING* at the end of an area version path is used to identify a staging area as the deployment area. Specifying an EDITION based deployment results in a comparison between the specified edition and the previous edition (using the iwcmp CLT) and the results used to determine which data records should be deployed. The criteria used to select records for deployment includes:
  - If the file is new or modified in the current area.
  - If the file has been deleted in from the current area
  - If the file is new and insert is performed in the WLCS database tables. If the file is modified, an update is performed in the WLCS database tables. If the file has been deleted, a deletion is made against the WLCS database table `wlcs_document`. This deletion cascades to the `wlcs_document_metadata` table. This is a limitation of the current reference implementation and will not likely remain in the product version of this interface.
- The `when` clause of the DNR specification must be defined as `client_after_deploy`. This results in the DNR being run on the client after a successful deployment.
- The use of correctly formatted version paths. The above example illustrates a valid version path for NT. An alternate version for NT is to use the file system interface. For example,

```
Y:\default\main\www\EDITION
```

For Unix, a version path can be specified using the following notation:

```
/iwmnt/default/main/www/EDITION
```

Refer to the TeamSite Command Line Tool reference manual for a discussion of version paths.

Completion of the above steps should result in a valid deployment.

Installing and Configuring Documentum

Perform the following steps to use Documentum with the BEA WebLogic Commerce Servers. Refer to Documentum's documentation for more information on setting up the Documentum Document Management System (DMS).

1. Add the `documentum.jar` to be deployed.
    - Add the following line to the EJB Deployment section in the `weblogic.properties` file:

```
/weblogicCommerce/ejb/documentum.jar,\
```

    - The `weblogic.properties` entry should now contain:

```
weblogic.ejb.deploy=\
    /weblogicCommerce/ejb/jar1.jar,\
    /weblogicCommerce/ejb/jar2.jar,\
    .
    .
    .
    /weblogicCommerce/ejb/documentum.jar,\
```

2. Modify the `CLASSPATH` to include Documentum's JDBC Driver classes and `Dfc.jar`.
    - Modify the `sethome.bat` (`sethome.sh` for UNIX) to set the following variables:
        - Set `DCTM_HOME` environment variable to the folder in which the Documentum classes and shared libraries are copied.
        - For NT, `set DCTM_HOME=%WL_COMMERCE_HOME%\documentum`
        - For UNIX, `DCTM_HOME=$WL_COMMERCE_HOME/documentum; export DCTM_HOME`
    - Add the following to the `WEBLOGIC_CLASSPATH`

    For NT, `%DCTM_HOME%\dfc.jar;%DCTM_HOME%`

    For UNIX, `$DCTM_HOME/dfc.jar:$DCTM_HOME`

3. Modify the PATH so that all the native shared Libraries required by the `Dfc.jar` are accessible.

   - For NT, `set PATH=%PATH%;%DCTM_HOME%`
   - For UNIX, add `LD_LIBRARY_PATH=$LD_LIBRARY_PATH;$DCTM_HOME; export LD_LIBRARY_PATH`

4. Set up the Environment Variable `DMCL_CONFIG` to indicate the location of the `dmcl.ini.`

   ```
   DMCL_CONFIG=%DCTM_HOME%\dmcl.ini
   ```

5. Modify the `dmcl.ini` file to point to the correct `DocBroker`.

   ```
    [DOCBROKER_PRIMARY]
    host = my_host
   ```

   Note: Please see the Documentum Server User's Guide for a list of parameters in the `dmcl.ini` and their purpose.

6. Create a connection pool for Documentum connections.

   - Modify `weblogic.properties` to create a connection Pool `dctmPool`

   ```
   weblogic.jdbc.connectionPool.dctmPool=\
          url=jdbc:documentum:oca,\
          driver=com.documentum.oca.jdbc.jdbc20.DjdbcDriver,\
          loginDelaySecs=1,\
          initialCapacity=1,\
          maxCapacity=5,\
          capacityIncrement=1,\
          allowShrinking=false,\
          shrinkPeriodMins=15,\
          refreshMinutes=10,\
   props=user=docbase_user;password=docbase_password;
       database=docbase_name;server=domain_name
   weblogic.allow.reserve.weblogic.jdbc.connectionPool.
       dctmPool=everyone
   ```

     - *`user`* is the Docbase User
     - *`password`* is the Docbase Password for that user.
     - *`database`* is the name of the DocBase

- `server` is the domain name, if required (applicable only for NT Docbases)

Documentum bean environment variables

The following are the environment properties used by the Documentum `Document` Bean and Documentum `Schema` Bean:

1. `DOC_TYPES` is of type `java.lang.String` and is a comma-separated list of all the Documentum Types that are available to be queried. This value is used in two instances:

   a. During creation of Rules users could select one of the types mentioned in the `DOC_TYPES` to create the rule.

   b. When the `DctmDocumentSmartBMP` receives a query, it does not know against which Object Types in the Docbase to query. So, the types mentioned in the value of `DOC_TYPES` would be queried in the order they appear in the comma-separated list. For example, if the Expression consists of a query `object_name == 'junk',` the `DctmDocumentSmartBMP` searches for the matching rows in the object type `newsitem` first, i.e.,

   ```
   select r_object_id from newsitem where object_name= 'junk'
   ```

   Then in promotion,

   ```
   select r_object_id from promotion where object_name= 'junk'
   ```

   and then in `dm_document`

   ```
   i.e. select r_object_id from dm_document where object_name=
   'junk'
   ```

   Note: One important thing to note here is that the results returned would be a union of all the results retrieved. The reason being as in this instance is there is a possibility of having both a type and its subtype in the `DOC_TYPES`. Here `promotion` and `newsitem` are both subtypes of `dm_document`.

   So all the results retrieved in the `select from newsitem` would be a subset of the results returned in the `select from dm_document` query.

Note: For more efficiency, if only a particular subtype is to be searched, include any additional attribute from that subtype in the Search Expression.

For example, If `newsitem` as a subtype of dm_document and has an attribute called `news_start_date` , include the `news_start_date` in the Search Expression. If the Expression consists of the query `object_name == ‘junk’ && news_start_date >= now,` then only the type newsitem would be queried for, because the attribute `news_start_date` is present only in the type `newsitem` from within all the types present in the variable `DOC_TYPES`.

2. `CONNECTION_URL` is of type `java.lang.String`, and the value is `jdbc20:weblogic:pool:dctmPool`, where `dctmPool` is the Connection Pool created for Documentum Sessions.
3. `SmartBMPClass` is of type `java.lang.String` and the value is `com.documentum.vendor.beasys.DctmDocumentSmartBMP`, the class `com.documentum.vendor.beasys.DctmDocumentSmartBMP` is Documentum’s `SmartBMP` Class to retrieve the `Content` from the `DocBase`.
4. `SmartBMPUpdate` is of type `java.lang.String` and defaults to `false`.
5. `SmartBMPLoadOnGetter` is of type `java.lang.String` and defaults to `false`.
6. `PropertyCase` is of type `java.lang.String` and defaults to `lower`. This is not required for the `Schema` bean.

The easiest way to configure the beans is to edit the `ejb-jar.xml` deployment descriptor contained in the `documentum.jar` from the install. After making the changes, recompile the EJBs in `documentum.jar` (i.e., run `ejbc`).

4 Example Applications

Example Applications includes the following topics:

Example Applications

Portal Reference Implementation Notes
- Login
- User/Group Creation
- Caching Properties
- Repository Considerations
- exampleportal Service Manager Entry

Example Applications

The example applications listed here work out of the box with the Cloudscape database and reference implementation DMS.

Note: Refer to "WebLogic Commerce Servers Configuration" on page 3-1 for information about making configuration changes to the system. You need to change configurations if you use a database other than Cloudscape or a DMS other than the reference implementation DMS shipped with WLCS 2.0.1. The configuration document also contains information about configuring other pieces of the system, including servlets and portals.

- The MyBuyBeans sample application is included as an example of the WebLogic Commerce Servers's power. The application demonstrates the commerce-based components of the system and the personalization services.

You can view the source in `public_html/buybeans` and `public_html/portals/buybeans`.

To understand how directory structures relate to the URLs, refer to “Setting up servlets and new portals” on page 3-8. To run the application, point your browser to `http://<wl-host>:<port>/mpbb`.

Note: `/mpbb` is an entry page for the MyBuyBeans portal that requires you to log in with an existing user. You can hit `http://<wl-host>:<port>/mybuybeans` to go directly to the MyBuyBeans portal without logging in and then create a new user with the tools provided in the upper-right corner of the page.

Note: `<port>` is the `ListenPort` of the running WebLogic Server set in the `weblogic.properties` file.

- View the Acme Portal example at `http://<wl-host>:<port>/exampleportal`. This portal demonstrates functionality of the Portal Management and User Management components.
- For personalization administration needs, point your browser to `http://<wl-host>:<port>/wlpsadmin` for the WebLogic Personalization Server administration tools. These tools allow you to create and manage property sets, users, rules, and portals.

Portal Reference Implementation Notes

A good way to get an overview of the new WebLogic Commerce Servers 2.0.1 Portal Management features is to examine the `exampleportal` reference implementation. The following sections explain various features of the implementation.

Login

Login to the reference implementation is achieved through the `<install-dir>/server/public_html/portals/repository/_userlogin.jsp` page. The following denotes how a user is validated and their portal, group, and user personalization are determined. This assumes that the user is on the login page, which may be accessed from the example portal.

1. Point your browser to `http://<wl-host>:<port>/exampleportal.`
2. Click the key icon in the upper-right corner of the page to log in.
3. Enter a user name and password.
4. Validate the user against the User Management realm. These are database tables specified in the `weblogic.properties` file.
5. Validate the Group Selection specified in the User Management administration tool.

- If the user belongs to only one group, the user will inherit the personalization attributes associated with that group.

Note: This group may or may not be associated with the portal. If it is associated with the portal they will get that groups personalization attributes. If the group is not associated with the portal, the group will inherit the default personalization attributes of the portal.

- If the user belongs to no groups (if created from the user management tools, and not associated with a group), the user will automatically inherit the personalization attributes of the default group specified in the portal registration (within weblogic.properties)
- If the user belongs to multiple groups, the user is presented with a list of groups from which to choose.

Note: The selected group may or may not be associated with the portal. If it is associated with the portal the user will inherit that group’s personalization attributes. If the group is not associated with the portal, the user will inherit the default personalization attributes of the portal.

- If you want to use the group association to the portal as a restrictive measure in addition to a personalization process, you will need to modify the _userlogin.jsp and _userreg.jsp pages to take this into account. The `<pt:getgroupsforportal>` returns a list of groups associated with the portal. It is up to the developer to cross reference these with the groups associated with the user for restrictive login purposes.

User/Group Creation

Default DB Realm

1. A user may be created through the user management tools if you're using the default realm. The user management tools also provide the option of attaching this user to a group.
2. If a new user is created from within the portal registration page (create your profile link), this will automatically add the user and then add the user to the AcmeUsers group.

Third Party Realms

1. A user or group may be created through implementation specific tools. For example, from a supported LDAP realm, you would use the tools that came with your LDAP server to manage your users and groups.
2. If using a third-party realm with the example portal, the registration page for creating users should not be used.

Caching Properties

The portal reference implementation has taken advantage of two primary caching schemes to improve performance.

1. User property information is retrieved only once during an active user's session. Be aware that if a login is in use on multiple browser instances, properties may be out of synchronization. For example, Scott is using login `acme`, and Tim also using login `acme`. They are both logged in concurrently. Scott changes acme's colors from green to red. Scott will see these changes immediately. Tim will not see these changes until he logs out and logs back in. To force a refresh on all properties each time, change the `usecache` attribute on the `<um:getproperty>` tag to `false` or leave the attribute out. Please refer to the `<um:getproperty>` tag information.

2. The displayed portlets are cached while the user is logged in or logged out. The portlet list is refreshed on a change in login status or when the session has been invalidated. However, the values are reset after a save within add/remove portlets and portlet layout. If you wish to have the list retrieved from the database on every display of the portal, edit the `<install-dir>/server/public_html/portals/repository/portalcontent.jsp`.

   - Uncomment the line:

```
Portlet[][] allMyPortlets = null;
```

   - Comment the line:

```
Portlet[][] allMyPortlets =
(Portlet[][])getSessionValue(PortalTagConstants.ALL_PORTLETS,
request)
```

Repository Considerations

The repository is a way to share files (images, portlets, headers, etc.) between multiple portals. The reference portal is shipped in the repository directory underneath the portals directory. If you want to create a portal, simply copy the repository directory structure to a new directory (do not copy the files within the directories). This new directory will become the working directory for your portal.

Note: Remember that you need to add the corresponding entry of the portal to the `weblogic.properties` file.

For example, the developer wishes to create a company specific version of the header for the new portal. Simply copy header.jsp to your new directory and modify it. During execution, the routing mechanism built into the portal framework checks the working directory for the JSP. If the file is not found, the framework retrieves the file from of the repository directory. So if you are creating portlets that will be shared across multiple portals, simply place them in `repository/portlets`. Two things to be aware of:

1. The refresh of the working directory files is controlled by the portal entry within `weblogic.properties`. The exampleportal entry is shipped with a 120 second refresh time. This means that the new working directory files will only be picked up every two minutes. A value of `-1`, means that every call routed through the portal will force a refresh on the working directory. This value is used for development purposes, mostly. For performance reasons, a value of 600 (5 minutes) or more should be used for deployment.
2. Any portlets placed within the working directory can not be shared across portals.

Setting up a portal

1. Use the Portal Management administration tool to create the portal.
2. Update the `weblogic.properties` file to include the new portal.
3. Create a new directory under `<install-dir>/server/public_html/portals` by copying the repository directory structure (excluding the files).
4. Restart the server.

exampleportal Service Manager Entry

This is a short reference explanation of the service manager entry for the example portal that ships with the product. For more information, please refer to the WLPS documentation.

Note: The **bold-faced text** in the entry comprises comments only. You should remove the bold text if you cut and paste this entry. If you leave it in, it will corrupt the `weblogic.properties` file.

```
weblogic.httpd.register.exampleportal=com.beasys.commerce.portal.
admin.PortalServiceManager
weblogic.httpd.initArgs.exampleportal=\ servlet name
    portalname=exampleportal,\ portal name defined within
        the tools
    homepage=/portals/example/portal.jsp,\ the home page for the
        portal
    defaultdest=/portals/example/portal.jsp,\ user is not
        logged in, where should the framework route
    workingdir=/portals/example/,\ where to check for
        overridden portal/portlet files
    groupname=AcmeUsers,\ default group for portal
    sessioncomparator=com.beasys.commerce.portal.admin.
        PortalSessionComparator,\ class to determine if the
        session is valid
    refreshworkingdir=120,\ how often to refresh the working
        directory entries
    repositorydir=/portals/repository/,\ where are the shared
        files
    timeout=99999,\ how long until the session and cookies timeout
    allowautologin=false if the cookies are valid, should the
        auto login feature be enabled
```

5 Migrating to WLCS 2.0.1

Migrating to WLCS 2.0.1 includes the following topics:

- Portal Data Migration
  - Portlet definition
  - Portal definition
  - Personalization
- Migrating Users
  - User Identification
  - Moving Users
  - Unique User Ids
- Migrating Groups
  - Group Identification
  - Moving Groups
  - Unique Group Ids
  - Group-User Hierarchies
- Properties and Property Sets
- JSP Migration
  - Sample Tables
  - Retired Tags
  - New Tags

Portal Data Migration

This is a basic overview for migrating your portal data from 1.7 to 2.0.1.

Portlet definition

You should recreate portlet definitions from scratch. Because of changes to key types within the portal framework, it's not practical to automatically migrate portlet definitions. Please follow the WLPS 2.0.1 documentation to create the new portlet definition.

Portal definition

You should recreate portal definitions from scratch. Because of changes to key types within the portal framework, it's not practical to automatically migrate portal definitions. Please follow the WLPS 2.0.1 documentation to create the new portal definition.

Personalization

There is no migration strategy for user- and group-personalized portals because no personalized information for users and groups will be lost.

Note: Only the user and group portal layouts will be lost, including personalized information such as background colors, etc. Refer to "Migrating Users" on page 5-3 and "Migrating Groups" on page 5-4 for information about handling the transfer of personalized information.

Migrating Users

User Identification

The notion of the system user has significantly changed in release 2.0.1. In the 1.7 release, a username could be repeated across multiple portals (qualified by a `<portalname>` prefix), and be used to represent different actual users. In this release, a user is identified by a single unique `String` identifier. The impetus for this shift in user name identification is the support of WebLogic security realms in release 2.0.1. Because in a security realm, such as an LDAP realm, a unique identifier represents one and only one system user, this pattern is mirrored in the Personalization Server. In the 1.7 release, a user did not exist outside the scope of a portal, users now exist independently of any portal.

Moving Users

User login and password information must be moved from the `PORTAL_SIGNON` table to the `WLCS_USER` table. The only information retained from table to table is the username and password. To this end, if a username (e.g. `bob`) is repeated in the `PORTAL_SIGNON` table, and is meant to represent two different users, a unique username (e.g. `bob2`, `bob3`) must be created for each redundant entry when making the transition.

It is strongly recommended that as these unique usernames are created, the `USER_HIERARCHY` table be updated to reflect the new usernames. The portal name/username combination can be used when seeking the usernames to update in the `USER_HIERARCHY` table.

If the desire is to no longer authenticate the users against the Personalization table, but against another realm, for example the LDAP realm, set the `IS_EXTERNAL` value for each transported user to 1. If the users are to be authenticated against the `WLCS_USER` table, using the `Realm` class `com.beasys.commerce.axiom.contact.security.RDBMSRealm`, set the `IS_EXTERNAL` value to 0.

Unique User Ids

Numeric user ids are used in portions of the Personalization Server to ensure strong performance and scalability. String identifiers are discarded in favor of long identifiers in places such as the portal framework user personalization tables and the group-user hierarchy table. Since a sequence table is used to generate unique ids for users as they are required, unique ids for users must be generated in the following manner:

For each user transitioned into the `WLCS_USER` table, retrieve the corresponding `com.beasys.commerce.axiom.contact.User` object and call its `getUniqueId` method.

This will place the following three entries in the `WLCS_ENTITY_ID_TABLE`:

- `JNDI_HOME_NAME`: `com.beasys.commerce.axiom.contact.User`
- `PK_STRING: <username>`
- `ENTITY_ID: <generated unique id>`

Migrating Groups

Group Identification

Just as users can now exist outside the scope of a portal, groups exist independently of portals, as well. Again, the impetus for this change is the support of WebLogic security realms by the WLPS 2.0.1 release.

Moving Groups

Group name information must be moved from the PORTAL_GROUP_HIERARCHY table to the WLCS_GROUP table. To this end, unique group names must be generated for each redundant group name entry in the PORTAL_GROUP_HIERARCHY table, and the group names (only) must be moved to the WLCS_GROUP table.

It is strongly recommended that as these unique usernames are created, the PORTAL_GROUP_HIERARCHY table be updated to reflect the new group names. The portal name/group name combination can be used when seeking the usernames to update in the PORTAL_GROUP_HIERARCHY table.

Unique Group Ids

Like user ids, group ids are shared as long values rather than String values. Since a sequence table is used to generate unique ids for groups as the ids are required, unique ids for users must be generated in the following manner:

For each user transitioned into the WLCS_GROUP table, retrieve the corresponding com.beasys.commerce.axiom.contact.Group object and call its getUniqueId method.

This will place the following three entries in the WLCS_ENTITY_TABLE:

- JNDI_HOME_NAME: com.beasys.commerce.axiom.contact.Group
- PK_STRING: <groupname>
- ENTITY_ID: <generated unique id>

Group-User Hierarchies

Once unique ids have been generated for users and groups, the transition of data into WLCS_GROUP_USER_HIERARCHY table can take place. For each entry in the PORTAL_USER_HIERARCHY, two unique numeric ids must be obtained: the id corresponding to the user name and the id corresponding to the group name. If the previous recommendations regarding updating the USER_HIERARCHY table were adhered to, obtaining the unique id for each entry should be straightforward. The WLCS_ENTITY_ID table contains the unique ids corresponding to the user and group names. These ids should be put into the appropriate columns of the WLCS_GROUP_USER_HIERARCHY table.

Properties and Property Sets

The properties in the WEBLOGIC_USER table pertaining to a particular portal can be translated into a single property set for subsequent portal use. To collect these properties, unique property names must be collected from the WEBLOGIC_USER table across user profiles where the username is like <portalname>. Once the property names are collected, a new PropertySet (com.beasys.commerce.foundation.property.Schema) object can be created, and the collected properties can be added to it. Because the 1.7 release dealt only with String properties, each new property type should be set as a String type. It is also recommended that each property be multi-valued and unrestricted, in order to maintain all property values. The new property set can be created programatically or through the use of the Property Set Management suite of administration tools.

It is strongly recommended that particular user and group property values are set programatically by obtaining each com.beasys.commerce.axiom.contact.User and com.beasys.commerce.axiom.contact.Group object, and setting properties for these objects through the ConfigurableEntity interface. The database schema for property values in release 2.0.1 is considerably more complex than the database schema of the 1.7 release.

JSP Migration

The real development effort of the customer is in creating and maintaining the JSP pages. There are several new enhancements with the 2.0.1 release. The exampleportal reference implementation takes advantage of these features (Please refer to the Portal

Management documentation and "Setting up a portal" on page 4-6 for more information). If you have not modified any of the pages for your own personal Web site, there will be no migration issues for the framework code. If any changes have been made, the following items need to be made clear.

1. All Portal JSPs should now use `<%@ page extends="com.beasys.commerce.portal.admin.PortalJspBase"%>` to extend the correct base class.
2. If you wish to use the repository features within 2.0.1, do not make fully qualified references to any pages that you wish to hit. For example when you want to forward to the login page (`_userlogin`), simply do:

```
<% setOverrideDestination(request, "_userlogin.jsp"); %>
<jsp:forward page="<%=getTrafficURI(request)%>"/>
```

The portal framework first looks for the file relative to the working directory (setup within the service manager registration). If the file is not there, it looks for the file relative to the repository directory (also setup within the service manager registration).

Note: If the files are fully qualified it will only look in the location specified.

Affected methods:

1. `setOverrideDestination` - now handles the repository.
2. `fixupRelativeURL` - now handles the repository.
3. any uses of `jsp:include` should now look something like:

```
<jsp:include page="<%=reconcileFile(request,
"alternateheader.jsp")%>"/>
reconcileFile will handle the repository aspects.
```

Sample Tables

If you are using the sample bookmarks and todo list tables, these may simply be renamed.

- Rename `TODO` to `WLCS_TODO`
- Rename `BOOKMARKS` to `WLCS_BOOKMARKS`

Retired Tags

Review the new tag documentation for the Personalization Server 2.0.1. Three tags that were in use in 1.7 have been retired.

1. `<pt:signon>` has been replaced by a combination of `<um:login>` and `PortalJspBase.setUser` and `PortalJspBase.setSuccessor`. `<um:login>` just checks to see if the user is valid. After validation, call `setUser` and `setSuccessor` to push the user and successor into the session. In general portal parlance, user is user and successor is group.
2. `<wl:sqlquery>` has been removed. Use `<es:preparedstatement>` to get the `ResultSet`.
3. `<pt:profile…>` tags have been removed. They have been replaced by several `<um>` tags.

New Tags

The tags discussed in this section are tags provided by the User Management subsystem of the Personalization Server that replace previously-used tags from the Portal tag library of release 1.7. Refer to the JSP Tag documentation for in-depth explanations of each tag.

<um:login>

The `<um:login>` tag replaces, in part, the `<pt:signon>` tag. The `<um:login>` tag provides username-password authentication against the security realm only. It performs no portal-particular activities, such as setting session values for the portal.

<um:getprofile>

The `<um:getprofile>` tag replaces, in part, the `<pt:profile>` tag. The `<um:getprofile>` tag retrieves a user/group profile for subsequent `setProperty` and `getProperty` calls.

<um:getproperty>

The `<um:getproperty>` tag replaces the `get` action of the `<pt:profile>` tag.

<um:setproperty>

The `<um:setproperty>` tag replaces the `set` action of the `<pt:profile>` tag. Unlike the old tag, the `<um:setproperty>` tag provides no profile `autocreate` option, as this option is no longer needed.

<um:removeproperty>

The `<um:removeproperty>` tag replaces the `remove` action of the `<pt:profile>` tag. Unlike the old tag, the `<um:removeproperty>` tag provides no profile `autodestroy` option, as this option is no longer needed.

<um:createuser>

The `<um:createuser>` tag replaces the `create` action of the `<pt:signon>` tag.