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

Migration Guide

Version 4.2
Document Date: April 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Migration Guide

Document Edition	Date	Software Versions
4.2	April 2002	BEA WebLogic Portal 4.0

Contents

3. Migrating WebLogic Commerce Server From Version 3.2 to 3.5

4. Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2

5. Migrating WebLogic Personalization Server From Version 2.0.1 to Version 3.1

Index

CHAPTER

1 Planning Your Migration

This chapter briefly outlines what you should expect from the migration process, before you begin, so that your migration can proceed as smoothly as possible.

- Migrating to 3.1.1, 3.2, or 3.5
- Migrating From 3.5 to 4.0
- Getting Supplementary and Updated Migration Documentation

Migrating to 3.1.1, 3.2, or 3.5

This section covers tips for migrating systems earlier than version 3.5.

- Before You Begin Migrating
- Manual Migration Process
- Getting Started

Before You Begin Migrating

Before you begin, make sure that you have completed all tasks related to licenses, installation, and database upgrades.

Warning: One of the most important factors in a successful migration is having the environment set up correctly. Be sure that you consult the documentation for all of the items in this section to be sure that your system meets the requirements for running the version you're migrating to.

Installation

Install the version that you want to migrate to; see the appropriate documentation for more information.

- Migrating to 3.1: `http://edocs.bea.com/wlcs/docs31/install/index.htm`
- Migrating to 3.2: `http://e-docs.bea.com/wlcs/docs32/install/index.htm`
- Migrating to 3.5: `http://e-docs.bea.com/wlcs/docs35/install/index.htm`

Note: The migration must be installed and run on a computer where WebLogic Server is installed. It uses information in the WLS weblogic.jar and the license file.

Be sure that you have installed the necessary Service Packs for your migration.

Supported Platforms

Ensure that your WebLogic Server version, operating system version, database version, and JDBC driver are supported by the version of WebLogic Portal that you want to upgrade to. See the appropriate documentation for more information on supported platforms.

- Versions in 3.1: `http://e-docs.bea.com/wlcs/docs31/relnotes/relnotes.htm`
- Versions in 3.2: `http://e-docs.bea.com/wlcs/docs32/relnotes/relnotes.htm`
- Versions in 3.5: `http://e-docs.bea.com/wlcs/docs35/relnotes/relnotes.htm`

Manual Migration Process

The migrations up to version 3.5 consist primarily of the following tasks:

- Reviewing all documentation for new and changed features

- Locating where in your code you use or depend on any of them
- Making the appropriate changes
- Running scripts, when included, to automatically migrate database schemas and some data

Be sure to review the documentation fully before beginning, then based on the changes and how they affect your code, planning an appropriate amount of time for migrating and verifying.

Migration Scripts for Migrations up to 3.5

Migration scripts for versions 2.01 to 3.5 are included in the `PORTAL_HOME\db\oracle\817\migration directory`. If you have not obtained the necessary scripts for those migration from another source, use the ones in that directory.

Getting Started

You cannot migrate directly to version 4.0. Complete all necessary migrations one at a time, in the order indicated. Be sure to plan adequate time if you are doing more than one migration; for instance, if you are currently on version 3.2 and want to migrate to version 4.0.

Table 1-1 Migrations to Complete

If you are currently running this version	Follow the directions in these chapters to migrate to 4.0
2.0.1	• Chapter 5, “Migrating WebLogic Personalization Server From Version 2.0.1 to Version 3.1” • Chapter 4, “Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2” • Chapter 3, “Migrating WebLogic Commerce Server From Version 3.2 to 3.5” • Chapter 2, “Migrating From WebLogic Commerce Server Version 3.5 to WebLogic Portal Version 4.0”
3.1.1	• Chapter 4, “Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2” • Chapter 3, “Migrating WebLogic Commerce Server From Version 3.2 to 3.5” • Chapter 2, “Migrating From WebLogic Commerce Server Version 3.5 to WebLogic Portal Version 4.0”
3.2	• Chapter 3, “Migrating WebLogic Commerce Server From Version 3.2 to 3.5” • Chapter 2, “Migrating From WebLogic Commerce Server Version 3.5 to WebLogic Portal Version 4.0”

Migrating From 3.5 to 4.0

This section contains the following information:

- Before You Begin Migrating
- Migration Tool Overview
- Getting Started

Before You Begin Migrating

Before you begin, make sure that you have completed all tasks related to licenses, installation, and database upgrades.

Warning: One of the most important factors in a successful migration is having the environment set up correctly. Be sure that you consult the documentation for all of the items in this section to be sure that your system meets the requirements for running the version you're migrating to.

Getting the Latest Migration Information

Check the support Web site for any updates to the migration process. Any updates will be described in a document titled "Migration Update" for the WebLogic Portal Release 4.0. The Web site is at `http://www.bea.com/support/index.jsp` You must have a valid support contract to get to this site.

Follow the directions in that document, and download any associated files listed in that document.

Note: Be sure that you have the latest version of the document and any associated files; if you downloaded them when you received this guide, download them and review them immediately before you begin migrating.

You should also download the latest version of this document, from `http://edocs.bea.com/`

Installation

Install the version that you want to migrate to; see the appropriate documentation for more information.

```
http://e-docs.bea.com/wlp/docs40/install/index.htm
```

Note: The migration must be installed and run on a computer where WebLogic Server is installed. It uses information in the WLS weblogic.jar and the license file.

Supported Platforms

Ensure that your WebLogic Server version, operating system version, database version, and JDBC driver are supported by the version of WebLogic Portal that you want to upgrade to. See the appropriate documentation for more information on supported platforms.

```
http://e-docs.bea.com/wlp/docs40/relnotes/relnotes.htm
```

You can migrate from a 3.5 installation, 3.5 with Service Pack 1, or 3.5 with Service Pack 2.

Migration Tool Overview

The 3.5 to 4.0 migration provides a migration tool that migrates code and data through the following tasks:

- Migrating your data to the new 4.0 format.
- Analyzes your code files and makes changes where possible, inserting an easily locatable comment in each file, where any change is made. If the tool cannot determine what change needs to be made, it will flag the code with a comment. You must then make the change to the code yourself. We strongly recommend that you review changes, whether made by the tool or simply flagged for your attention.

Note: You will still need to perform an additional number of migration tasks manually.

For more extensive information about what the tool migrates, see “Getting Started With the Migration Tool” on page 2-2.

Getting Started

Continue to the section titled “Getting Started With the Migration Tool” on page 2-2, within Chapter 2, “Migrating From WebLogic Commerce Server Version 3.5 to WebLogic Portal Version 4.0.”

Getting Supplementary and Updated Migration Documentation

Regardless of the version you are updating to, complete these steps before you begin migrating.

- If you are reading a local copy of this document, get the latest version online at `http://edocs.bea.com/wlp/docs40/pmigrate/index.htm`
- Go to the BEA Developers’ Center to check for any recent migration information or files: `http://developer.bea.com`
- Go to the BEA support site to check for any recent migration information or files: `http://www.bea.com/support/index.jsp`

Note: You must have a valid support contract to get to the support site.

CHAPTER

2 Migrating From WebLogic Commerce Server Version 3.5 to WebLogic Portal Version 4.0

This chapter addresses the 4.0 changes to Campaign Manager for WebLogic, and WebLogic Commerce Server and WebLogic Personalization Server.

This section includes the following topics. Read "Getting Started With the Migration Tool" thoroughly to determine what steps to follow and the information you need, then continue through this chapter as directed in the migration checklist.

- Getting Started With the Migration Tool
- Migrating Data From 3.5 to 4.0
- Migrating Code From 3.5 to 4.0
- Moving New and Migrated Data Files to Your 4.0 Directory
- Manual Migration Tasks for 3.5 to 4.0
- Verifying the Migration From 3.5 to 4.0
- Migration Reference Information
- Viewing Code Changes Using the Migration Viewer Tool

Getting Started With the Migration Tool

This section covers the following:

- Migration Checklist for 3.5 to 4.0
- How the Migration Tool Works
- Pre-Migration Steps

Migration Checklist for 3.5 to 4.0

1. Be sure you've already reviewed the information in Chapter 1, "Planning Your Migration," beginning on page 1-1.
2. Check the support Web site for any updates to the migration process. The Web site is at `http://www.bea.com/support/index.jsp` You must have a valid support contract to get to this site.

   Follow the directions in that document, and download any associated files listed in that document.

Note: Be sure that you have the latest version; if you downloaded it when you received this guide, download the document again and review it immediately before you begin migrating.

3. Go to `http:\\edocs.bea.com` and download or view the latest version of this document.
4. We recommend that you review "Migration Reference Information" on page 2-71 before you begin migration procedures. A topic within that section, "How to Use This Information" on page 2-72, describes why you need the information and how to put it to use.
5. Review this section, "Getting Started With the Migration Tool" on page 2-2, to understand how the migration tool, new in this release, will migrate your files.
6. Follow the instructions in "Migrating Data From 3.5 to 4.0" on page 2-16 to migrate your databases.

7. Then continue to the "Migrating Code From 3.5 to 4.0" on page 2-45 for instructions on migrating code. This section contains detailed information about new items you might want to use, as well as changes to make to existing code.
8. Follow the steps in "Manual Migration Tasks for 3.5 to 4.0" on page 2-55 to migrate aspects of your implementation that could not be migrated by the tool.
9. Follow the steps in "Verifying the Migration From 3.5 to 4.0" on page 2-69 to verify that the migration has been completed correctly.

How the Migration Tool Works

Review this section for information about the new migration tool.

- Migration Tool Overview
- What the Tool Migrates
- Helper Files in the Migration Directory

Migration Tool Overview

A migration tool is included in this release to assist in migrating your code and data to this release. It is not a J2EE application and does not need any connection to WebLogic Server; it is a stand-alone tool dealing with the file system.

Note: It is not a full migration application: it is a *helper utility* that evaluates code files and databases, migrates where possible, and copies the new data to a destination location. However, you will need to do a significant amount of updating yourself.

The tool has two components, a code migrator and a data migrator. Each migrates your code and data to a destination directory you specify.

Note: Migrate your *data* to a destination directory that is not associated with your 4.0 directory. Be sure that all migratable data is successfully migrated before copying it to your 4.0 directory. (Instructions for copying it to the appropriate directories are included in "Moving New and Migrated Data Files to Your 4.0 Directory" on page 2-56.)

Code Migration Process Overview

The code migrator analyzes `.java` files and either makes the appropriate change or flags the necessary portions for change, without updating the code. No matter what the change, a comment is added to the file. The code migrator assists you in the migration process, but you must still review all code files before you can consider the migration complete.

The amount of code it can migrate varies drastically from one implementation to another.

The changes and comments are put in a new copy of the files, created in a location you specify before beginning the migration.

You can migrate both standard `.java` files and JSPs, by compiling the JSPs, running the code migration tool, and making the appropriate updates to JSPs, tags, and any other dependent code.

The code migration tool focuses on the following:

- Import statements (packages and fully qualified classes)
- Variable declarations
- Variable instantiations
- Variable references
- Method calls
- Field references
- Castings
- Instanceofs

At the beginning of code migration, you specify a source (3.5) directory and migration destination directory, which are typically your WLCS3.5 `src` directory and your Portal 4.0 `src` directory.

It is ultimately your responsibility to actually work through the comments in the migrated code and make some modifications yourself. In addition, if you want to incorporate new features in this release described in this section, you must write that code yourself, as well.

Data Migration Process Overview

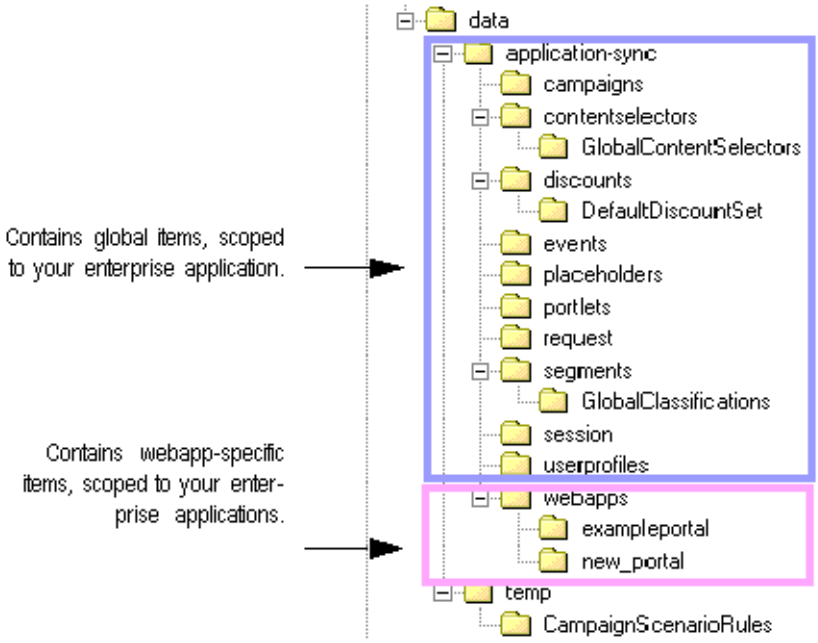
The data migrator transfers your existing database data, including rulesets, to the new database schema and XML files. It will typically migrate a high percentage of your data, though this varies from one implementation to another.

The migration process performs a variety of tasks, including leaving some files as is, renaming files, creating new files, and so on. The database migration takes place on the 3.5 tables themselves; it does not create a new set of 4.0 tables without touching the 3.5 tables.

- Any 3.5 tables that are not used or changed in 4.0 is left as is; drop them after you have been up and running on 4.0 for a few weeks.
- Any 3.5 tables that are renamed or otherwise changed, are modified and become 4.0 tables.

Before beginning, you specify a source (3.5) directory and a migration destination directory. The new XML files are created in the destination directory, shown in Figure 2-1, which you will later move to the Portal 4.0 directory.

Figure 2-1 Data Migration Destination Directory



What the Tool Migrates

The migration tool will make many of the changes for you; however, you must review all the changes to your code and data, and you will need to make some changes manually. Therefore, we strongly recommend that you review this section to be well informed when you review migrator tool changes and make manual changes. In addition, the tool does not migrate all aspects of your site. Table 2-1 lists main components of a typical implementation and what is and is not migrated.

Table 2-1 Components Migrated and Not Migrated

Component	Migrated?	Comments
Java code, including JSP tags	Yes, to your data migration source directory	The migrator might not be able to migrate all code; it will flag all code it cannot migrate, and you need to make the necessary changes.

Table 2-1 Components Migrated and Not Migrated

Component	Migrated?	Comments
JSPs	Not directly	However, we suggest that you generate the.`java` files for the JSPs and run the code migrator on the results. Then update your JSPs accordingly.
Portals, including portlet colors, layouts, and so on4.0:JSPs, whether migrated	Partial, to the data destination directory.	Portlets are migrated. In addition, minimal HTML files are created defining each of your portals and its portlets. See "Re-creating Portals and Portlets" on page 2-67 for information on manual migration steps. Portals and portlets might be renamed during migration; see "Naming and Renaming Rules for Placeholders and Other Items" on page 2-91.
Database schemas and data	Yes, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).	Some tables have been removed in this release and the data is stored in XML files, instead.
Rules, campaigns, scenarios	Yes, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).	Rules are migrated first. Any rules that are *not* independent of scenarios and campaigns are assumed to be scenario rules. Next, campaigns and scenarios are migrated, and scenarios are embedded in their associated campaigns. The previously migrated scenario rules are then brought into the migrated campaigns, as well. Rules might be renamed during migration; see "Naming and Renaming Rules for Placeholders and Other Items" on page 2-91.
Events that you can view in the EBCC that are defined using the property set schema, the `.evt` files.	Yes, to the data destination directory.	

Table 2-1 Components Migrated and Not Migrated

Component	Migrated?	Comments
Events stored in the database (unlinked to the events defined and viewed through EBCC)	No	See "Migrating Events" on page 2-63 for information on manual migration steps.
Webflow and Pipeline	Partial, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).	Code files and data are migrated; however, the structure has changed significantly in this release. See "Migrating Webflow and Pipeline" on page 2-61 for information on manual migration steps.
webflow.properties, pipeline.properties	Yes, to data_dest_dir\main.wf and main.pln	
weblogiccommerce.properties	Yes; to two application.xml and application-config.xml	"Entering Information Formerly Stored in weblogiccommerce.properties" on page 2-60.
Deployment descriptors, web.xml, etc.	No	See "Migrating Deployment Descriptors and Other Configuration Files" on page 2-59.
Licenses	No	Refer to the 4.0 installation documentation.
Discounts	Yes, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).	Discounts are now stored in XML files, one file per discount.
Placeholders	Yes, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).	Placeholders are now stored as XML files, one file per placeholder. Placeholders are also likely to be renamed during migration according to the restrictions in "Naming and Renaming Rules for Placeholders and Other Items" on page 2-91.

Table 2-1 Components Migrated and Not Migrated

Component	Migrated?	Comments
Configurable entities and property sets	Yes, in the 3.5 database (databases) and to the data destination directory (XML files derived from database data).*	*Some user-defined configurable entity successor values must be migrated manually. See “Migrating non-User and non-Group Configurable Entity Successor Property Values” on page 2-62.

Helper Files in the Migration Directory

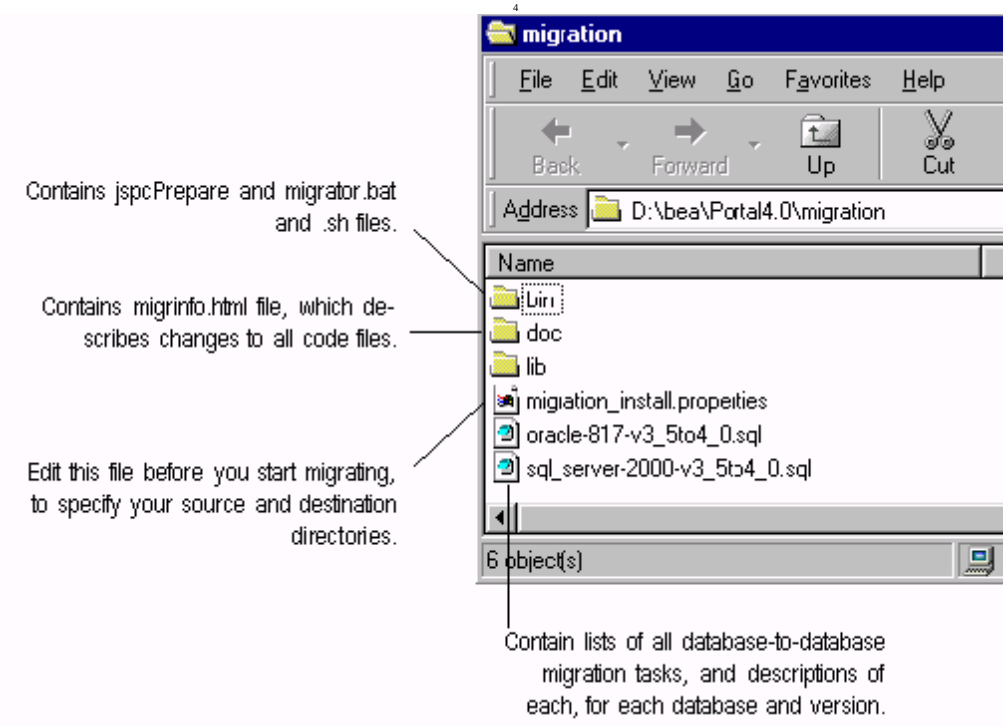
The migration directory contains the programs and support files for the migration process.

- Migration Directory Structure
- Migration Log File
- Reference Information Files for Code Changes in This Release
- .SQL Files for Customized Database Conversion

Migration Directory Structure

Figure shows the `migration` directory.

Figure 2-2 4.0 Migration Directory



Migration Log File

The tool writes information to a `migration. log` file in the `migration` directory, recording the actions that take place. The log file also includes system properties information such as tool versions, service packs installed, and other migration environment information, Use this file if you encounter difficulties or have questions while using the migration tool.

Reference Information Files for Code Changes in This Release

The key changes to code in this release, such as packages and whether code has been removed or deprecated, is documented in the `migrinfo`.html file, included in the `doc` directory within `migration`. The information in these files is used by the code

migration tool to add comments to each file it migrates or flags for additional review. However, we recommend that you review this information now, particularly the information on classes or packages that are key to your site implementation.

Note: You can view, sort, and search the information using the Migration Viewer tool. See "Viewing Code Changes Using the Migration Viewer Tool" on page 2-105.

.SQL Files for Customized Database Conversion

All the SQL used in the data migration to migrate 3.5 databases to 4.0 databases, for each database and version, is included in the `migration` directory. There is one file for each type and version of database; for instance, one file is named `oracle-817-v3_5to4_0.sql`. If you made modifications to your databases, you will need to use these files to create the appropriate modified SQL statements and run them. Instructions are included in "Migrating Databases You Have Changed" on page 2-42.

Note: These files do not contain code that migrates 3.5 databases to 4.0 XML files.

Pre-Migration Steps

Complete these steps to ensure that your system is prepared correctly to begin the migration.

- Check Release Notes for Main Release and Service Packs
- Starting BEA WebLogic Portal 4.0
- Making a Complete Backup
- Getting the Latest Migration Files
- Checking Migration Environment
- Editing the 4.0 set-environment bat file and migration_install.properties File

Check Release Notes for Main Release and Service Packs

The release notes for any released Service Packs provide important information. Review them before completing a migration.

Starting BEA WebLogic Portal 4.0

Follow the instructions in the *Installation Guide* and any other referenced documentation in that guide to get release 4.0 running on your system. Be sure that your installation environment and the software are running correctly before you continue.

Set up your 4.0 build environment system—creating .jar files, etc.—so that you will be able to modify, test, and deploy converted code files later in the migration process.

Making a Complete Backup

Completely back up your entire 3.5 system at least once.

Getting the Latest Migration Files

Check the support Web site for any updates to the migration process. The Web site is at `http://www.bea.com/support/index.jsp` You must have a valid support contract to get to this site.

Follow the directions in that document, and download any associated files listed in that document.

Note: Be sure that you have the latest version of the document and any associated files; if you downloaded them when you received this guide, download them and review them immediately before you begin migrating.

If you are not reading this online, download the latest version of this document now from `http://edocs.bea.com/wlp/docs40`

Checking Migration Environment

Be sure that the migration files were installed on a computer where WebLogic Server is installed. The migration requires access to the `weblogic.jar` and license files in the BEA directory.

Editing the 4.0 set-environment bat file and migration_install.properties File

Before you begin, you need to enter configuration settings.

1. Open the set-environment.bat file for your 4.0 installation. Comment out all databases you are not using, and remove comments in front of the database you are using.
2. Open the `PORTAL_HOME\migration\migration_install.properties` file. Listing 2-1 following this procedure shows the file. Items you need to address are in bold.
3. Set the line `migration_start` equal to `true`, instead of `false`.
4. Enter the source and data destination directory for the migration. The source directory is your WLCS35 directory; the data destination directory should be in a different location, such as `D:\\migration40` . (Use double slashes in the path.) The XML files for 4.0 will be created in a directory structure at that location.
5. Specify the enterprise application name that will be applied to data that is application scoped in 4.0, on the line `application_name = wlcsApp`. You need to do this if you want your current data to be scoped to an application with another name. Do not change this property once you start migration.

```
#application_name = wlcsApp
```

Note: Most data in this release, such as campaigns and placeholders, is enterprise application scoped. It is common to all Web applications within the enterprise application. If you have two or more enterprise applications, you will be prompted to address this issue later in "Moving New and Migrated Data Files to Your 4.0 Directory" on page 2-56.

6. Change the bold values in Listing 2-1, such as your ID, password, database, and database driver, to the appropriate values for your system.

   For the value `server=YOURSERVER`, enter the name of the database server. For instance, for Oracle, enter the Oracle SID.
7. Comment out (add `#`) the databases you are not using, and remove the comment mark in front of the database you are using.
8. The `database.continue.despite.failure` setting is primarily useful if you encounter errors during the data migration and need to run it again. Setting the value to `true` will allow you to continue past errors and get more information about the error.

9. If you are using **SQL Server** or **Sybase**, set to `true` a line in the file that controls whether you can use `java.sql.Clob` objects for CLOB (character large object) database column retrieval. This is because some databases don't support using `ResultSet.getString()` for CLOB retrieval, which is the default behavior.

   The line is as follows: if `false`, the database is using `getClob`() and `setClob`() methods. If `true`, the database is using `getString`() and `SetString`() methods.

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

Listing 2-1 Partial migration_install.properties File

```
#####################################################################
## PROPERTIES TO BE SET BY THE USER AT 'INSTALL' TIME
#####################################################################
# ------------------------------------------------------------
##       When you have set the properties you need, set the
#       following flag to 'true' so the migrator tool will start.
#       Otherwise, the Migrator will assume you have NOT set
#       any property and will refuse to run.
## ------------------------------------------------------------
start_migrator=false   (Set this to true.)
# ------------------------------------------------------------
##      Moving Data from 'data_src_dir' if not from database
#       (for webflow.properties and pipeline.properties in 3.5),
#       to 'migr_dst_dir', where 4.0 files will be created,
#       such as .usr, .pln, etc.
## ------------------------------------------------------------
#
# General migration settings
# The directory separator is \\ (espaced \). For instance,
# data_src_dir = d:\\thisdir\\thatdir\\srcdir
#
data_src_dir = d:\\bea\\WLCS35     (Your 3.5 root directory)
migr_dst_dir = d:\\40migration     (A separate directory.)
# The application name that will be applied to data that is
# application scoped in 4.0
# Edit this default name if you want your current data
# to be scoped to an application with another name.
```

```
# This property should not be changed once migration has started.
#
application_name = wlcsApp      (The name of your enterprise application.)
# ----------------------------------------------------------------
## Database Properties
## ----------------------------------------------------------------
# Database connection properties
#
# These are some hardcoded WebLogic jDriver for Oracle 8.1.5 settings
# until this file can be generated at build / install time
#
database.connection.driver = weblogic.jdbc.oci.Driver
database.connection.url = jdbc:weblogic:oracle
database.connection.props =
user=YOURUSER;password=YOURPASSWORD;server=YOURSERVER;weblogic.t3.waitForConnec
tion=true;weblogic.t3.waitSecondsForConnection=999999999999;weblogic.jts.waitSe
condsForConnectionSecs=999999999999
#
# Database and version
#
database.name=oracle    (Comment out the versions you do not use.)
database.version=817
#database.name=sql_server
#database.version=2000
#database.name=db2
#database.version=7
#database.name=sybase
#database.version=12
#
# This is used to continue a task even if required statements fail.
# Use it to get a longer list of failed actions
# (You might want to set it to true if you have already solved, but cannot
# implement the solution yet, for the error that is halting migration.)
#
database.continue.despite.failure=false  (Set to true to continue past errors.)
#
# 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 (Set to true to allow use of
```

ResultSet.getString().)

Migrating Data From 3.5 to 4.0

This section covers information on how the migrator tool works and how to complete the migration tasks.

- Before You Begin
- Migrating Your Data

Before You Begin

Review this information and complete tasks, if necessary, before using the migration tool.

- How the Data Migration Works
- Descriptions of Data Migration Tool Tasks
- Specifying Unneeded Migration Tasks
- Removing Anonymous Constraints

How the Data Migration Works

Note: For an overview of what data is converted during migration, see "Data Migration Process Overview" on page 2-5. For a list of the table changes between 3.5 and 4.0, see "Database Changes" on page 2-73. The separate migration tasks and corresponding descriptions are included later in this section in "Descriptions of Data Migration Tool Tasks" on page 2-17.

The data migration process relies primarily on the following properties files, `rdbms-migration-task.properties` at and multiple versions of `rdbms-migration-sql.properties`. The files are stored in the `PORTAL_HOME\migration\lib\migration.jar` file and unjarred at the appropriate time during migration.

Examples are shown as Table 2-2 and Table 2-3.

Table 2-2 Example of entries in rdbms-migration-task.properties file

Task number	Name of SQL statement
1.001	generate_portal_report.PORTAL_P13N

Table 2-3 Example of entries in rdbms-migration-sql.properties file

Name of SQL statement	SQL statement
generate_portal_report.PORTAL_P13N	*(SQL code)*

The migration process iterates through the tasks and runs the associated SQL. The tasks are listed in "Descriptions of Data Migration Tool Tasks" on page 2-17.

Information about the data migration is written to the `migration.log` file in the `PORTAL_HOME\migration` directory and is displayed in the migration tool window.

No 3.5 tables are dropped during the process, to ensure the greatest possible data integrity. Once you have verified that the migration is successful and you have been running on the new version for a week or more, you might choose to drop unused 3.5 tables. Use your SQL tools to do so.

Descriptions of Data Migration Tool Tasks

This section describes the tasks that the tool completes and what is involved in each. You do not have to read this section before you migrate data, but you will need to refer to it if any of the tasks are not run successfully.

Note: Pay particular attention to the dependencies among these tasks, and to whether they can be repeated without restoring from a backup.

Webflow Migration

In release 3.5, a Webflow is defined by a property file, `webflow.properties`, and is system scoped. That means that all applications relying on a Webflow, such a commerce applications, get one configuration. Editing the `webflow.properties` file is done via a JSP-based tree structure.

In release 4.0, a Webflow is Web application scoped, defined in a `.wf` file. In fact, many `.wf` files may compose a Webflow. These files do not reside under the root directory as in release 4.0; they reside under the Web application's EBCC directory structure, so as to be deployed with the Web application, via Data Sync. These `.wf` files are XML files, not properties files.

The role of migration of Webflow is to read the `webflow.properties` under the root directory of a release 3.5 installation, transform it (twice) using XSL to create an XML `main.wf` file that can be used to start a Web application-specific Webflow using the EBCC Webflow Editor.

Validation is done after both transformation steps to verify the final `main.wf` file follows the Webflow XSD definition.

Dependencies on earlier tasks: None.

Pipeline Migration

The Pipeline migration follows closely the Webflow migration, at least conceptually. The release 3.5 root directory, system-scoped `pipeline.properties` file is transformed into a `.pln` file, `main.pln`, after two transformations are applied via XSL.

Do Webflow and Pipeline migration together, or not at all.

Dependencies on earlier tasks: None.

Property Set/Schema Migration

In release 3.5, a set of tables represents the schema/property sets:

- WLCS_SCHEMA: The overall description of a schema, with a name, a group, and a SCHEMA_ID
- WLCS_PROP_MD: For a given schema, identified by the SCHEMA_ID, lists all the properties that compose the schema and the meta-data of these properties.
- WLCS_PROP_MD_BOOLEAN through WLCS_PROP_MD_USER_DEFINED contain typed values, either the default property value for that schema, or a list of possible choices in the schema (see Design Details later in this task description).

The migration of Schema/Property Sets takes the data in these tables, transforms the data into XML files following the EBCC directory structure, using XSL as the main engine to create the structure of the XML files. These files' extensions also follow the EBCC file naming conventions.

Dependencies on earlier tasks: None.

Design Details

To migrate the Schema/Property sets, the migration executes the following SQL statements:

1. `select * from WLCS_SCHEMA ;` this statement provides the list of all files to create, and their name, directory structure, and extensions. Using the result set from 1, the migration will execute:

2. `select * from WLCS_PROP_MD where SCHEMA_ID='?';` this statement provides the list of properties to fill in for each selected schema. Based on the PROPERTY_TYPE value of this result set, the migration will execute a statement:

3. `select * from WLCS_PROP_MD_TEXT WHERE PROPERTY_META_DATA_ID = '?';` this statement provides the list of restricted properties values to fill in for each selected property, as well as the default value based on the "IS_DEFAULT' value.

As the result sets become available, the migration builds the XML file directly, using Strings, writes it to the appropriate location, and validates it. There is no XSL needed with this approach.

The migration also tracks the number of file types written and to what location, in order to show that information as the localized message at the end of the processing.

WLCS_SCHEMA

This table is used to derive the overall <ps:propertyset> "header", with <ps:name>, <ps:description>, and <ps:type>

- SCHEMA_GROUP_NAME: contains the type of property to be migrated. The ps:type node (under ps:propertySet) is set to that value. If the schema is not part of the table below (content, for instance), it is not migrated. The following is a summary of the structures:

Table 2-4 File Extensions

Userprofiles	*.usr
Events	*.evt
Catalog	*.clg
Request	*.req
Session	*.ses

- SCOPE_NAME: The name of the schema, and will become the file prefix and the <ps:name> value of the migrated file (as per the table above)
- DESCRIPTION: The value of this column will be put in the <ps:description> node.
- SCHEMA_ID: will be used to tie other pieces together.

WLCS_PROP_MD

This table is used to derive the overall <ps:property> list under each <ps:propertyset>, together with most of the values for the sub-nodes <ps-name>, <ps-description>, <ps-ismultivalued>, <ps:is-restricted>, <ps:datatype>

- PROPERTY_NAME: will be used to fill in the <ps:name>
- DESCRIPTION: will be used to fill in the <ps:description>
- IS_RESCTRICTED: <ps:is-restricted>, if true, then the <ps:restricted-value> nodes need to be put in the <ps:property> node, based on the PROPERTY_TYPE and its corresponding table, WLCS_PROP_MD_<type>. If IS_RESTRICTED is false, there is no need to go to any WLCS_PROP_MD table, and the <ps:property> node will not contain any <ps:restricted-value> node. However, the WCLS_PROP_MD_<type> might still contain the default value based on the IS_DEFAULT column value.
- IS_MULTIVALUED: <ps-ismultivalued>
- PROPERTY_TYPE: <ps:datatype>
- PROPERTY_META_DATA_ID: the link to the actual values, possibly restricted or default, based on the IS_DEFAULT column in the corresponding

```
WLCS_MD_PROP_<type>:select * from  WLCS_PROP_MD_TEXT where
PROPERTY_META_DATA_ID='155';
```

Table 2-5 Type Value and Corresponding Table

WLCS_PROP_MD. PROPERTY_TYPE	Corresponding Table
0	WLCS_PROP_MD_BOOLEAN
1	WLCS_PROP_MD_INTEGER
2	WLCS_PROP_MD_FLOAT
3	WLCS_PROP_MD_TEXT
4	WLCS_PROP_MD_DATETIME
5	User_Defined: Not ported.
6	Multi-valued: idem

Portal Reports

The migration process generates reports about portal data that you can use to re-create your portals after the data migration process is complete. The reports are created in the following directory: *`migration_dest_dir`*`\data\PortalReport`

Note: The file uses 0 and 1 to indicate whether an item is included in a portal; 0 is no, 1 is yes.

That location is also displayed in the migration tool window, when you perform the Portal Reports data migration task. You will be prompted to use the portal reports to re-create your portals in “Re-creating Portals and Portlets” on page 2-67. Instructions for using the portal reports to re-create your portals are included in “Re-creating Portals and Portlets” on page 2-67.

WEBLOGIC_IS_ALIVE RDBMS Migration

Creates new 4.0 WEBLOGIC_IS_ALIVE tables. Includes the following steps:

- drop_40_table.WEBLOGIC_IS_ALIVE
- create_40_table.WEBLOGIC_IS_ALIVE

- copy_to_40_table.WEBLOGIC_IS_ALIVE

Dependencies on earlier tasks: None.

SEQUENCER RDBMS Migration

Creates new 4.0 SEQUENCER table and loads it with 3.5 WLCS_SEQUENCER data. Includes the following steps:

- drop_40_table.SEQUENCER
- create_40_table.SEQUENCER
- copy_to_40_table.SEQUENCER

Dependencies on earlier tasks: None.

ENTITY RDBMS Migration

Creates new 4.0 ENTITY tables and migrates 3.5 WLCS_ENTITY data where the JNDI_HOME_NAME is either "`com.beasys.commerce.axiom.contact.User`" or "`com.beasys.commerce.axiom.contact.Group`". Includes the following steps:

- drop_40_table.ENTITY
- create_40_table.ENTITY
- copy_to_40_table.ENTITY.user
- copy_to_40_table.ENTITY.group

Dependencies on earlier tasks: None.

PROPERTY RDBMS Migration

Creates new 4.0 PROPERTY_KEY and PROPERTY_VALUE tables and loads them with transformed 3.5 WLCS_PROPERTY table data. Includes the following steps:

- drop_40_table.PROPERTY_KEY
- create_40_table.PROPERTY_KEY
- drop_35_table_temp.PROPERTY_KEY_TEMP35
- create_35_table_temp.PROPERTY_KEY_TEMP35

- copy_to_35_table_temp.PROPERTY_KEY_TEMP35
- update_35_table_temp.PROPERTY_KEY_TEMP35.property_key_id
- copy_to_40_table.PROPERTY_KEY

Dependencies on earlier tasks: None.

CATALOG PROPERTY RDBMS Migration

Creates new 4.0 CATALOG_PROPERTY_KEY and CATALOG_PROPERTY_VALUE tables and loads them with transformed 3.5 WLCS_CATALOG_PROPERTY table data.

Dependencies on earlier tasks: None.

USER_SECURITY RDBMS Migration

Creates new 4.0 USER_SECURITY loads it with the transformed 3.5 WLCS_USER and WLCS_ENTITY table data. Includes the following steps:

- drop_40_table.USER_SECURITY
- create_40_table.USER_SECURITY
- copy_to_40_table.USER_SECURITY
- drop_35_table_temp.MAX_ENTITY_ID
- create_35_table_temp.MAX_ENTITY_ID
- copy_to_35_table_temp.MAX_ENTITY_ID
- drop_35_table_temp.USER_SECURITY_TEMP35
- create_35_table_temp.USER_SECURITY_TEMP35
- copy_to_35_table_temp.USER_SECURITY_TEMP35
- copy_to_40_table.USER_SECURITY

Dependencies on earlier tasks: None.

GROUP_SECURITY RDBMS Migration

Creates new 4.0 GROUP_SECURITY table and loads it with transformed 3.5 WLCS_GROUP and WLCS_ENTITY_ID table data. Includes the following steps:

- drop_40_table.GROUP_SECURITY
- create_40_table.GROUP_SECURITY
- copy_to_40_table.GROUP_SECURITY

Dependencies on earlier tasks: None.

USER_GROUP_CACHE RDBMS Migration

Creates new 4.0 USER_GROUP_CACHE table and loads it with 3.5 WLCS_USER_GOUP_CACHE table data. Includes the following steps:

- drop_40_table.USER_GROUP_CACHE
- create_40_table.USER_GROUP_CACHE
- copy_to_40_table.USER_GROUP_CACHE

Dependencies on earlier tasks: None.

GROUP_HIERARCHY RDBMS Migration

Creates new 4.0 GROUP_HIERARCHY table and loads it with transformed 3.5 WLCS_GROUP_HIERARCHY table data. Includes the following steps:

- drop_40_table.GROUP_HIERARCHY
- create_40_table.GROUP_HIERARCHY
- copy_to_40_table.GROUP_HIERARCHY

Dependencies on earlier tasks: None.

USER_GROUP_HIERARCHY RDBMS Migration

Creates new 4.0 USER_GROUP_HIERARCHY table and loads it with transformed 3.5 WLCS_USER_GROUP_HIERARCHY table data. Includes the following steps:

- drop_40_table.USER_GROUP_HIERARCHY

- create_40_table.USER_GROUP_HIERARCHY
- copy_to_40_table.USER_GROUP_HIERARCHY

Dependencies on earlier tasks: None.

DOCUMENT RDBMS Migration

Creates new 4.0 DOCUMENT and DOCUMENT_METADATA tables and loads them with transformed 3.5 WLCS_DOCUMENT and WLCS_DOCUMENT_METADATA table data. Includes the following steps:

- drop_40_table.DOCUMENT
- create_40_table.DOCUMENT
- copy_to_40_table.DOCUMENT

Dependencies on earlier tasks: None.

DISCOUNT RDBMS Migration

Creates new 4.0 DISCOUNT table in preparation for Discount Migration. Includes the following steps:

- drop_35_fk_constraint.DISCOUNT_ASSOCIATION.fk2_discount_association
- drop_35_fk_constraint.ORDER_ADJUSTMENT.fk1_order_adjustment
- drop_35_fk_constraint.ORDER_LINE_ADJUSTMENT.fk1_order_line_adjustment
- drop_temporary_table.DISCOUNT_TEMP35
- create_temporary_table.DISCOUNT_TEMP35
- copy_to_temporary_table.DISCOUNT_TEMP35
- drop_35_table.DISCOUNT_TEMP40
- create_40_table.DISCOUNT_TEMP40
- copy_to_40_table.DISCOUNT_TEMP40
- drop_35_table.DISCOUNT
- create_40_table.DISCOUNT

- copy_to_40_table.DISCOUNT

Note: This task is not repeatable without first restoring 3.5 tables from backup.

Dependencies on earlier tasks: None.

SCENARIO_END_STATE RDBMS Migration

Creates new 4.0 SCENARIO_END_STATE table and loads it with transformed 3.5 SCENARIO_END_STATE table data.

Three columns have been renamed:

- SCENARIO_ID NUMERIC(15) to SCENARIO_XML_REF VARCHAR2(254)
- USER_ID VARCHAR2(50) to USER_NAME VARCHAR2(200)
- CONTAINER_ID VARCHAR2(50) to CONTAINER_REF VARCHAR2(254)

Includes the following steps:

- drop_35_table_temp.SCENARIO_END_STATE_TEMP35
- create_35_table_temp.SCENARIO_END_STATE_TEMP35
- drop_40_table_temp.SCENARIO_END_STATE_TEMP40
- create_40_table_temp.SCENARIO_END_STATE_TEMP40
- copy_to_40_table_temp.SCENARIO_END_STATE_TEMP40
- drop_35_table.SCENARIO_END_STATE
- create_40_table.SCENARIO_END_STATE

Dependencies on earlier tasks: None.

USER_PROFILE RDBMS Migration

Creates new 4.0 USER_PROFILE table and loads it with transformed 3.5 WLCS_USER_PROFILE table data. Includes the following steps:

- drop_40_table.USER_PROFILE

- create_40_table.USER_PROFILE
- copy_to_40_table.USER_PROFILE

Dependencies on earlier tasks: None.

AD_BUCKET RDBMS Migration

Creates new 4.0 AD_BUCKET table in preparation for Campaign / Placeholder Clean Up. Includes the following steps:

Note: New column: MODIFIED_DATE DATE DEFAULT SYSDATE

- USER_ID renamed to USER_NAME
- PLACEHOLDER_NAME renamed to PLACEHOLDER_XML_REF
- CONTEXT_UID renamed to CONTEXT_REF
- CONTAINER_UID renamed to CONTAINER_REF
- NEW COLUMN: APPLICATION_NAME

Note: This task is not repeatable without first restoring 3.5 tables from backup.

Dependencies on earlier tasks: None.

AD_COUNT RDBMS Migration

Creates new 4.0 AD_COUNT table in preparation for Campaign / Placeholder Clean Up.

Table modifications.

- AD_IDENTIFIER renamed to AD_ID
- CONTAINER_UID renamed to CONTAINER_REF
- NEW COLUMN: APPLICATION_NAME

Includes the following steps:

- drop_40_table_temp.AD_COUNT_TEMP40
- create_40_table_temp.AD_COUNT_TEMP40

- copy_to_40_table_temp.AD_COUNT_TEMP40
- drop_35_table.AD_COUNT
- create_40_table.AD_COUNT

Dependencies on earlier tasks: None.

MAIL_BATCH RDBMS Migration

Creates new 4.0 MAIL_BATCH table and loads it with 3.5 MAIL_BATCH data.

Note: This task is not repeatable without first restoring 3.5 tables from backup.

Dependencies on earlier tasks: None.

ORDER_LINE_ADJUSTMENT RDBMS Migration

Modify 3.5 ORDER_LINE_ADJUSTMENT table to 4.0 format.

Note: This task is not repeatable without first restoring 3.5 tables from backup.

Dependencies on earlier tasks: None.

Create New Tables

Creates new 4.0 tables:

- DATA_SYNC_APPLICATION
- DATA_SYNC_ITEM
- DATA_SYNC_SCHEMA_URI
- DATA_SYNC_VERSION
- ENTITLEMENT_RULESET
- LAYOUT
- PORTAL
- PORTAL_P13N
- RESOURCE_GROUP_ADMIN

- PORTAL_P13N_SKIN_POOL
- PORTAL_PAGE
- PORTAL_PAGE_P13N
- PORTAL_PAGE_P13N_LAYOUT_POOL
- PORTLET
- PORTLET_P13N
- PORTAL_P13N_LAYOUT
- PORTLET_PLACEHOLDER
- SKIN

No changes to the following:

- MAIL_ADDRESS
- MAIL_BATCH
- MAIL_BATCH_ENTRY
- MAIL_HEADER
- MAIL_MESSAGE
- PLACEHOLDER_PREVIEW
- WLCS_CATEGORY
- WLCS_PRODUCT
- WLCS_PRODUCT_CATEGORY
- WLCS_PRODUCT_KEYWORD
- WLCS_CUSTOMER
- WLCS_SHIPPING_ADDRESS
- WLCS_CREDIT_CARD
- WLCS_TRANSACTION
- WLCS_TRANSACTION_ENTRY

- WLCS_SAVED_ITEM_LIST
- WLCS_ORDER
- WLCS_ORDER_LINE
- WLCS_SHIPPING_METHOD
- WLCS_SECURITY
- DISCOUNT_ASSOCIATION
- ORDER_ADJUSTMENT
- ORDER_LINE_ADJUSTMENT

Dependencies on earlier tasks: None.

Discount Migration

The release 3.5 discount set XML documents are retrieved from the DISCOUNT_SET table. The documents are then XSL-transformed into the release 4.0 format using `discounts.xsl`. In release 4.0, the EBCC expects discounts to be stored one discount per file. The XML tag `<discountSet>` is no longer supported. After the original discount set documents have been transformed to the new format, the document(s) are split into individual Discount documents. The discount documents are then validated against the discount new schema and any validation failures are reported to the user. Whether validation succeeds or not, the resulting discount documents are saved to:

`<migr_dst_dir>/data/application-sync/discounts/<original_set_name>/`

Documents that failed validation are saved to aid in trouble shooting the problem. Each discount file has an encoded version of the value from the original discount's `<name>` tag and a `.dis` file extension. The discount's `<name>` tag is updated to match the file name.

Finally the DISCOUNT_NAME and APPLICATION_NAME fields of the release 4.0 version of the DISCOUNT table are updated. An earlier task should have partially migrated the data already.

Dependencies on earlier tasks: The final part of this task, the update of records in the DISCOUNT table, depends on the successful completion of the DISCOUNT RDBMS Migration task. The first part of the task, where the XML definitions of the discounts are generated will execute correctly even if the DISCOUNT RDBMS Migration task has not been run.

Rules Migration

The release 3.5 rule set documents are retrieved from the RULESET table. The default/GlobalCatalogSelectors set is ignored if it is still in the database. This rule set was part of the release 3.5 sample data but is was never documented or used by any services or samples in release 3.5. Each Rule Set document is then XSL transformed into the release 4.0 format using `ruleSet.xsl`. The transformed documents are validated against the release 4.0 schemas.

The EBCC in release 4.0 expects rules to be stored one rule per document so the rule sets are split apart. Each rule file name is an encoded version of the original rule's `<name>` tag value. The rule's `<name>` tag is updated to match the file name. This applies to the "default/GlobalClassifications" and "default/GlobalContentSelectors" rule sets only. The only other rule sets supported by the release 3.5 EBCC were rule sets used in scenarios. Therefore any rule sets other the "default/GlobalClassifications" and "default/GlobalContentSelectors" sets found in the RULESET table are assumed to be for scenarios and are stored as rule sets in a temporary location for use during Campaign migration.

The new rule sets are saved to the following locations:

- default/GlobalClassifications are saved to

  `<migr_dst_dir>/data/application-sync/segments/GlobalClassifications/` with a '.seg' extension.

- default/GlobalContentSelectors are saved to

  `<migr_dst_dir>/data/application-sync/contentselectors/GlobalContentSelectors/` with a '.sel' extension.

- All others are assumed to be Scenario Rule Sets and are saved to a temporary location for use in the Campaign Migration

  `<migr_dst_dir>/data/temp/CampaignScenarioRules`

Note that the rule set documents will be saved even if they fail to validate to aid in troubleshooting.

Dependencies on earlier tasks: None.

Campaign Migration

The release 3.5 campaign documents are retrieved from the CAMPAIGN table and the release 3.5 Scenario documents from the SCENARIO table. The Campaign document(s) are XSL transformed into the release 4.0 format using "`campaign.xsl`". Likewise, the scenario document(s) are XSL transformed into the release 4.0 format using "`scenario.xsl`".

In release 4.0 the EBCC expects campaign documents to be a single compound document. That is, each campaign has its own XML document. Each campaign contains one or more scenarios. The scenarios were stored as separate documents in release 3.5 and specified by a link node in the campaign. In release 4.0 scenario(s) are merged into the campaign. In turn, each scenario contains one or more rule sets. Release 4.0 still uses a link node to specify the GlobalClassifications rule set but scenario specific rules (the ones saved to the temporary location in the rule set migration task) are merged into the scenario.

Data from the CAMPAIGN_SCENARIO table, along with the campaign scenario link tags, are used to determine which scenarios get merged into which campaigns and where. The CAMPAIGN_SCENARIO table data is used to identify the scenarios used by a given campaign. The release 3.5 campaign documents specify linked scenarios using the <`scenario-link` > tag. For example:

```
<scenario-link scenario-id="TourCampaign1_scenario1" />
```

The value of the scenario-id attribute is the SCENARIO_UID column entry in the SCENARIO table. This tag is used to determine which scenario is placed at a particular location in the campaign document.

Next, rule sets are merged into the campaign documents; that is, into the scenarios that use them. In release 3.5 rule sets were linked to scenarios using the scenario-ruleset-link tag. For example:

```
<scenario-ruleset-link ruleset-id="TourCampaign2_scenario1_rules" />.
```

The value of the `ruleset-id` attribute matches the name of a scenario rule set from the RULESET table (it actually matches the part of the name after the "default/" prefix). Note that these rules were converted to the release 4.0 format, then stored as rule sets in the temporary location (`<migr_dst_dir>/data/temp/CampaignScenarioRules`) during the Rule Migration task.

Once the complete compound campaign documents have been generated, a final xsl transformation is performed to clean up a few things in the documents. At this point, the campaign documents are in their final form so they are then validated against the release 4.0 campaign schema. Finally, they are saved to the file system at the following location:

`<migr_dst_dir>/data/application-sync/campaigns/`

using the original CAMPAIGN_UID from the CAMPAIGN table and a `.cam` extension. Note that the campaign documents will be saved even if they fail to validate to aid in troubleshooting.

Dependencies on earlier tasks: The Rules Migration task; it migrates the scenario rule documents that are then used in the campaign documents.

Placeholder Migration

The release 3.5 placeholder documents are retrieved from the PLACEHOLDER table. They already exist as individual documents and remain so in release 4.0. The documents are then updated to the release 4.0 format using an XSL transformation; the `xsl` file is called `placeholder.xsl`. The transformed documents are then validated against the release 4.0 schema. Placeholders are saved to:

`<migr_dst_dir>/data/application-sync/placeholders/`

The Placeholder's file name is generated by encoding the value in the PLACEHOLDER_NAME column in the PLACEHOLDER table. The Placeholder document's `<name>` tag is updated to match the generated file name. Placeholder files have a `.pla` extension. Note that the placeholder documents will be saved even if they fail to validate to aid in troubleshooting.

Dependencies on earlier tasks: None.

Campaign / Placeholder Clean Up

This task performs data updates only on the AD_BUCKET, AD_COUNT, and SCENARIO_END_STATE tables. These tables have already been created and temporary copies of them populated with some version of their data by an earlier "RDBMS" task. When this task begins it expects to find the partially migrated data for each table in a temporary table with the release 4.0 schema. These tables have a _TEMP40 suffix on their names, that is AD_BUCKET_TEMP40, AD_COUNT_TEMP40, and SCENARIO_END_STATE_TEMP40. Then for each table the following operations are performed.

1. `DELETE FROM <release 4.0 production table>` – In case the task has already been run and the target production table already has data.
2. `INSERT INTO <release 4.0 production table> SELECT FROM <_TEMP40 table>` – To move the partially migrated data into the production table.
3. An update statement specific to the table. See the following for details for each table.

The AD_COUNT table references to campaigns (the CONTAINER_REF column's values) are updated so the values are the URL of the campaigns. The AD_BUCKET table references to campaigns, the CONTAINER_REF column's values, Scenarios, the CONTEXT_REF column's values and placeholders, the PLACEHOLDER_XML_REF column's values are updated so each is a URL to the document. The SCENARIO_END_STATE table references to campaigns are the CONTAINER_REF column's values and scenarios. The SCENARIO_XML_REF column's values are updated so each is a URL to the document. Also, for each table the APPLICATION_NAME column is updated with a user-definable default enterprise application name. The default name is `wlcsApp`, which is defined in the `migration_install.properties` file by the property named `application_name`.

Dependencies on earlier tasks: SCENARIO_END_STATE RDBMS Migration, AD_BUCKET RDBMS Migration, and AD_COUNT RDBMS Migration. These tasks step up the tables and data that this task operates on.

Successor Property Value Migration

This is the final step in property migration. Successor property values had to be handled separate from the rest of property migration because of the way the successor property value is stored. In release 3.5 a Configurable Entity could have a property named `_successor\u007e` whose value is a BLOB. In release 4.0 the value of a successor property is a long; that is, the ENTITY_ID of the entity that is the successor. The BLOB is a serialized instance of a class called `PersistableHandle`. The `PersistableHandle` is de-serialized into a runtime object and contains the `SmartKey` and a `String`, the JNDI home name, of the successor. The `SmartKey` is cast to a specific type of entity bean primary key class and an identifier is obtained. The identifier and the JNDI home name are used to lookup the ENTITY_ID of the successor in the ENTITY table. The PROPERTY_VALUE table is then updated, replacing the BLOB value of the successor with the long value.

Note: The migration tool can only support entities with JNDI home names of `com.beasys.commerce.axiom.contact.Group` and `com.beasys.commerce.axiom.contact.User`. The migrator only knows the primary key classes for these types, you would have defined any others.

Dependencies on earlier tasks: PROPERTY RDBMS Migration task. It performs the migration of all other properties and copies the successor property values into the PROPERTY_VALUE_TEMP40 table where this task expects to find them. Again the _TEMP40 table is used to allow this task to be repeated. Values are obtained from the PROPERTY_VALUE_TEMP40 table and updates are made to the PROPERTY_VALUE table.

Portal / Portlet Migration

Due to the extensive changes to portals from the release 3.5 to the 4.0 releases, the portal migration is a partial migration that requires the user to take the migration output files and complete the migration using the EBCC to create a new portal layout. Portlets are, however, completed migrated. The XML definition files created from the existing portlet data are complete and can be used without further alteration.

Note: See also "Webflow Migration" on page 2-17.

First, portlet data is retrieved from the WLCS_PORTLET_DEFINITION table and a generic XML representation of the data is created. The document structure is a simple representation of the rows returned from the table. For example:

```
<?xml version="1.0" ?>
<result-set>
  <row>
    <column name="NAME" value="Defined Portlets" />
    <column name="CONTENT_URL" value="portlets/definedportlets.jsp" />
    <column name="HEADER_URL" value="##NULL##" />
    <column name="ALTERNATE_HEADER_URL" value="##NULL##" />
    <column name="FOOTER_URL" value="##NULL##" />
    <column name="ALTERNATE_FOOTER_URL" value="##NULL##" />
    <column name="TITLEBAR_URL" value="titlebar.jsp" />
    <column name="BANNER_URL" value="##NULL##" />
    <column name="EDITABLE" value="0" />
    <column name="EDIT_URL" value="##NULL##" />
    <column name="HELP" value="0" />
    <column name="HELP_URL" value="##NULL##" />
    <column name="ICON_URL" value="##NULL##" />
```

```
    <column name="MINIMIZEABLE" value="1" />
    <column name="MAXIMIZEABLE" value="1" />
    <column name="MAXIMIZED_URL" value="##NULL##" />
    <column name="MANDATORY" value="0" />
    <column name="MOVEABLE" value="1" />
    <column name="FLOATABLE" value="1" />
    <column name="MINIMIZED" value="0" />
    <column name="LOGIN_REQUIRED" value="0" />
  </row>
… any number of more rows…
</result-set>
```

Note that this is strictly internal representation of the portlet data. This file is never written to the file system. It is shown here strictly to help you understand the XSL transformation file used to produce the final document.

Next, the generic portlet document is XSL transformed into the new release 4.0 format using "`portletGenToCompound.xsl`". The transformed document is split so each portlet is in its own document. The portlet documents are validated against the new schema.

Finally, they are saved to `<migr_dst_dir>/data/application-sync/portlets/`

The portlet file name is an encoded version of the value of the portlet-name tag and the file has a `.portlet` extension. The portlet's `portlet-name` tag value is updated to match the file name.

Next, portals are migrated. Portal data is retrieved from the WLCS_PORTAL_DEFINITION table and into a generic XML format, the same structure as defined above for portlets. Again, this file is never written to the file system. The generic portal document is transformed into the new release 4.0 format using `portalGenToCompund.xsl`. References to the portlet(s) that each portal contains based on data from WLCS_PORTAL_HIERARCHY table are then added to each portal document. The portal documents are split so each portal is in its own document. They are saved to:

`<migr_dst_dir>/data/application-sync/webapps/<portal_name>/`

The portal file name is an encoded version of the value of the NAME column of the WLCS_PORTAL_DEFINITION and the file has a `.portal` extension. The portal files are templates that can be used as a starting point to create a new version of the portal using the EBCC. They cannot be validated against the schema at this point.

Dependencies on earlier tasks: None.

Drop Existing Constraints

Drops 3.5 constraints.

Dependencies on earlier tasks: If you are using a database other than Oracle, you must manually drop anonymous constraints ***after*** the Portal / Portlet Migration task and ***immediately before*** you complete this task, Drop Existing Constraints. Use the database tool of your choice to check your database for anonymous constraints and remove them.

Create New Constraints

Creates new 4.0 constraints.

Dependencies on earlier tasks: None.

Specifying Unneeded Migration Tasks

By default, all migration tasks are required, or "non-skippable." This is to ensure that all the elements that a migration task depends on have been completed. For example, if task B only works if task A has created a new 4.0 table, task A needs to be done for task B to work. You cannot select a task in the migrator window unless all of the migration tasks above it in the list have been completed.

We recommend that you complete all tasks. However, if you have not implemented some features, you can make them skippable. For instance, if you do not use have any existing portals, you do not need to run related tasks, including the portal reports.

Making the task skippable means that you do not have to complete it before you complete other tasks below it in the list of tasks in the migration window.

To make a task skippable, follow these steps.

1. Refer to the previous section, "Descriptions of Data Migration Tool Tasks" on page 2-17 to determine the name of the task you want to make skippable.
2. Locate the `PORTAL_HOME\migration\lib\migration.jar` file and unjar it. The `DataMigratorBundle.properties` file is one of the files that will result, in the `PORTAL_HOME\migration\lib\com\bea\commerce\migration\tools` directory. Open it for edit. The file contains each task and corresponding information; the placeholders task is shown in the following example.

Listing 2-2 Partial Contents of `DataMigratorBundle.properties` file

```
#
# The placeholders task
#

placeholders_title=Placeholder Migration

placeholders_description=Moves the placeholder definitions out of the data
base,from the PLACEHOLDER table in the 3.5 format, to the new 4.0 format.

You must perform this task for proper migration.

placeholders_classname=com.bea.commerce.migration.data.version.v3_5to4_0.PlaceH
olderMigration

placeholders_gif=null

placeholders_skippable=false
```

3. For each task, locate the `*_skippable` property at the end of the task information and change its value from `false` to `true`. Save the file and close it.

Removing Anonymous Constraints

If you are using Oracle, skip this section.

If you are using a database other than Oracle, you must manually drop anonymous constraints ***after*** the **Portal / Portlet Migration** task and ***immediately before*** you complete the task **Drop Existing Constraints**. . Use the database tool of your choice to check your database for anonymous constraints and remove them.

Migrating Your Data

Follow the steps in this section to complete data migration steps with the migration tool.

For an overview of the process, see “Data Migration Process Overview” on page 2-5.

Be sure you have edited the `migration_install.properties` file in the `PORTAL_HOME\migration` directory. If you have not entered the appropriate information, an error message will appear when you attempt to run the migrator.

You might want too use a database viewing/editing tool to check the data migration process as it progresses. After each task that affects databases, you can view your 3.5 database and the changes that took place.

1. Shut down WebLogic Commerce Server and Personalization Server.
2. Make *two* complete backups of your data, including both the databases and your properties files. Include the behavior tracking/events database, if it is being used.
3. Start the migration tool by double-clicking or using the command line to run the `migrator.bat` or `migrator.sh` file in the `PORTAL_HOME\migration\bin` directory.

Note: If you encounter problems, edit the file and be sure all settings are correct.

Listing 2-4 shows the migrator.bat file.

Listing 2-3 Example migrator.bat File

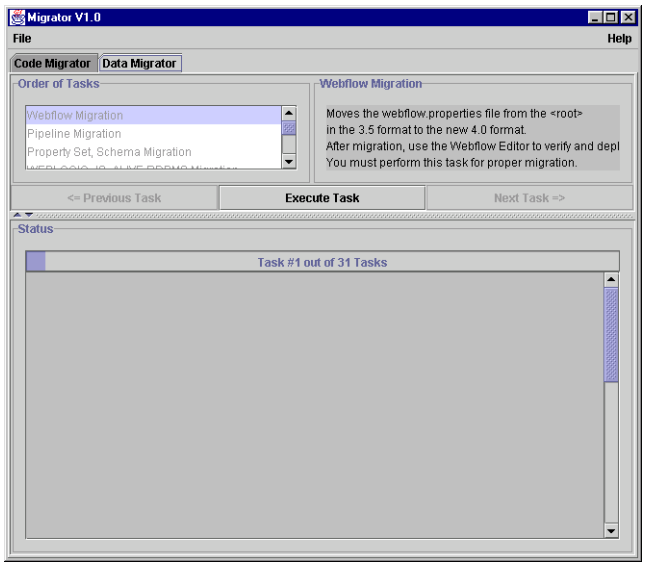
```
set JDK_1-3_BIN=%BEA_HOME%\jdk131\bin
set MIGRATION_DIR=%WL_COMMERCE_HOME%\migration
set MIGRATION_LIB=%MIGRATION_DIR%\lib
set
MIG_CLASSPATH=%BEA_HOME%\lib\tools.jar;%MIGRATION_LIB%\migration.jar;%MIGRATION
_LIB%\wlcs3_5classes.jar;%MIGRATION_LIB%\apache\xerces-1_3_1\xerces.jar;%MIGRAT
ION_LIB%\apache\xalan-j_2_0_1\xalan.jar;%WEBLOGIC_HOME%\lib\weblogic.jar;%BEA_H
OME%

%JDK_1-3_BIN%\java -cp %MIG_CLASSPATH%  -DMIGRATION_DIR=%migration_dir%
com.bea.commerce.migration.tools.Migrator

echo on
```

4. Click the tab labeled Data Migrator, if it is not already on top, as shown in Figure 2-3.

Figure 2-3 Data Migration Window



5. Select the first task in the list and click Execute Task. Status messages will appear, showing the progress of the migration.

   Note on messages in the data migration window: Many of the messages are extremely detailed, and contain valuable information, sometimes indicating the directory where a migrated or new file was created. Tables affected by the migration task are listed. At the bottom of each set of comments for the task, a message states whether the migration was successful. For some tasks, messages will state that a task could not be completed; this is typically not an issue because of service packs that you have installed that have already completed the task, or because the task is optional. The `migration.log` file also contains all messages displayed in the window.

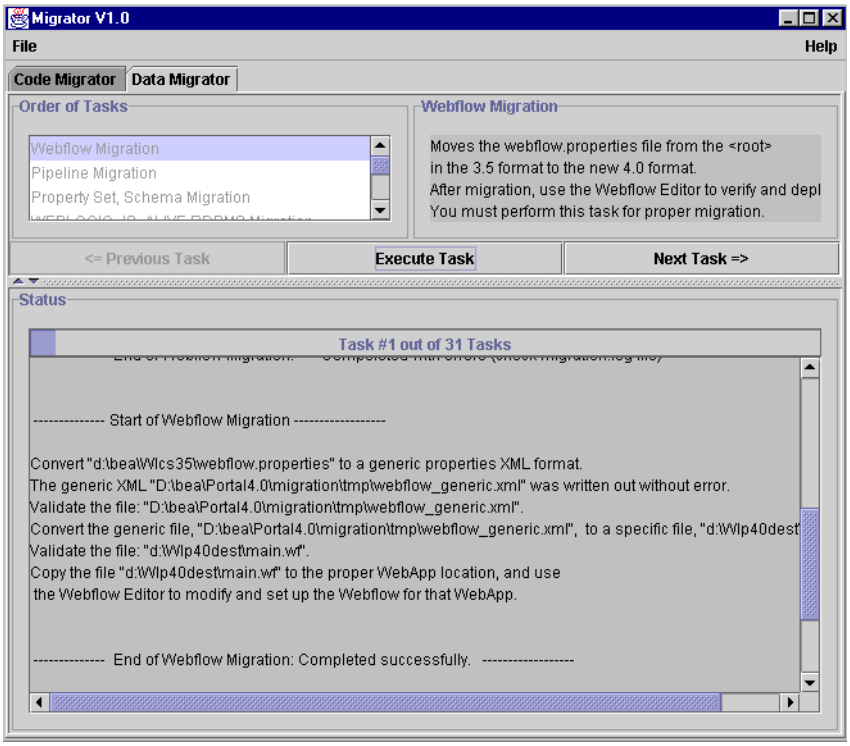
   If any other message appears stating that there were problems migrating, use the error messages onscreen to address the problem, then run the task again. (Some tasks require that you restore the database from a backup, then fix the problem and run the task again. If this is the case, the description in the window says so.) If the problem stems from modifications you have made to the database, see "Migrating Databases You Have Changed" on page 2-42.

   Note on the Drop Existing Constraints task: If you are using Oracle, skip this section. If you are using a database other than Oracle, you must manually drop anonymous constraints ***after*** the **Portal / Portlet Migration** task

and ***before*** you complete the task **Drop Existing Constraints**. Use the database tool of your choice to check your database for anonymous constraints and remove them.

Figure 2-4 shows the comments for successful migration of the Webflow Migration task.

Figure 2-4 Data Migration Completion and Comments



6. When it has completed successfully, click the Next Task button, and continue until all tasks are completed. (If you have made a task skippable, you can skip it in the list; otherwise, you must migrate every task in sequence.)

Manual Migration for Previously Customized Items

If you have modified databases, or modified or extended JNDI_HOME_NAME or PROPERTY_SET, complete the appropriate tasks in this section.

- Migrating Databases You Have Changed
- Migrating JNDI_HOME_NAME and PROPERTY_SET

Migrating Databases You Have Changed

If you changed the definition of any of your 3.5 databases, such as adding a column or changing a column name, you need to run those migrations manually. Use the appropriate `.sql` file in the `PORTAL_HOME\migration` directory to get the SQL you need to modify. There is one file for each type and version of database.

Follow these steps to migrate your changed databases.

1. List all the database changes you have made.
2. Read the data migration descriptions and dependencies in the "Descriptions of Data Migration Tool Tasks" on page 2-17 to determine which tasks are affected by your database changes.
3. Locate the appropriate `.sql` file for your database; they are stored in the `PORTAL_HOME\migration` directory.
4. Locate the tasks you identified previously. Each set of SQL is preceded by a commented line and the task name, as shown. The SQL contains everything you need except for terminate statements. You will need to add the appropriate termination, such as a semicolon, when you run the SQL yourself.

```
#  ---------------------
# SEQUENCER RDBMS Migration
DROP TABLE SEQUENCER
CREATE TABLE SEQUENCER ( SEQUENCE_NAME VARCHAR2(50) NOT NULL, CURRENT_VALUE
NUMBER(15) NOT NULL, IS_LOCKED NUMBER(1) NOT NULL)
INSERT INTO SEQUENCER ( SEQUENCE_NAME, CURRENT_VALUE, IS_LOCKED) SELECT
SEQUENCE_NAME, CURRENT_VALUE, IS_LOCKED FROM WLCS_SEQUENCER
```

5. For each identified migration task, create new SQL statements, editing the SQL as appropriate for your database changes.

6. Run the SQL statements on the database using the SQL tool of your choice, such as SQLPlus.

Migrating JNDI_HOME_NAME and PROPERTY_SET

If you created your own JNDI_HOME_NAME, PROPERTY_SET, or both, you will need to run the tasks referencing those values again, to convert the items you created. The rows for the migrated information will be added to the 4.0 tables or files. The affected tables are ENTITY, PROPERTY_KEY, CATALOG_ENTITY, CATALOG_PROPERTY_KEY, CATALOG_PROPERTY_VALUE, and PROPERTY_VALUE.

Complete the following steps for each of the tasks listed here.

- ENTITY RDBMS Migration
- PROPERTY RDBMS Migration
- CATALOG PROPERTY RDBMS Migration
- USER_SECURITY RDBMS Migration
- GROUP_SECURITY RDBMS Migration

1. Locate the `*.sql` file in the `PORTAL_HOME\migration` directory for your database and database version, such as `oracle-817-v3_5to4_0.sql`.
2. Locate the first task, ENTITY RDBMS Migration, in the `.sql` file. Partial code for that task is shown in Figure 2-5.

Figure 2-5 Partial SQL File for Task Containing JNDI_HOME_NAME

```
-- ENTITY RDBMS Migration

DROP TABLE ENTITY

...

INSERT INTO ENTITY ( ENTITY_ID, ENTITY_NAME, ENTITY_TYPE, CREATION_DATE,
MODIFIED_DATE) SELECT ENTITY_ID, PK_STRING, 'User', GETDATE() - 90, GETDATE() FROM
WLCS_ENTITY_ID WHERE JNDI_HOME_NAME = 'com.beasys.commerce.axiom.contact.User'

...
```

3. In the SQL for that task, find the items in Table 2-6and modify the values appropriately wherever they occur.

You need to supply the appropriate entity type, JNDI_HOME_NAME, and property set type for each new version you have created. Locate the values you need to change in the SQL for each task by searching for all of the following and making the appropriate replacement.

Table 2-6 Entities and Related Items to Change in SQL

Search for this	Comments	Replace it with this
`'User' following "PK_STRING"`	This is an entity type that the migrator was set up to look for and migrate. It can be the same as the JNDI_HOME_NAME. (See note later in this procedure for a caution regarding the names.)	The name you assigned to your entity type.
`'Group' following "PK_STRING"`	This is the other entity type that the migrator was set up to look for and migrate. It can be the same as the JNDI_HOME_NAME. (See note later in this procedure for a caution regarding the names.)	The name you assigned to your entity type.
`WHERE_JNDI_HOME_NAME = 'User'`	The JNDI home name for the entity type referenced in the same SQL block, corresponding to the entity type you specify for the *first* line in this table.	The JNDI home name for the entity type.
`WHERE_JNDI_HOME_NAME = 'Group'`	The JNDI home name for the entity type referenced in the same SQL block, corresponding to the entity type you specify for the *second* line in this table.	The JNDI home name for the entity type.
`PK.PROPERTY_SET_TYPE = 'USER'`		The value from your 3.5 SCHEMA_GROUP_NAME in the WLCS_SCHEMA table.

4. Run the SQL statement once for each new entity type you have created.

Note: You can modify the SQL to create all new entity types at once; however, you will need to modify the SQL appropriately to do so. If you run the SQL with the references to home names as is, the entity type and JNDI home name *must* be exactly the same.

When to Drop 3.5 Tables

The migration process does not drop 3.5 tables when it creates 4.0 table and XML files. We strongly recommend that you not drop any 3.5 tables until you have been running your site successfully in 4.0 for at least a week. See "Verifying the Migration From 3.5 to 4.0" on page 2-69 for suggestions on how to verify that the migration was successful.

Migrating Code From 3.5 to 4.0

Use the information in this section to migrate the code. The information in this section is organized as follows:

- Before You Begin
- Migrating Code
- Migrating JSPs

Before You Begin

This section provides helpful information about steps to complete before migrating.

- Excluding Unmodified BEA Example Files From the Migration Process
- Team Migration Guidelines

Excluding Unmodified BEA Example Files From the Migration Process

You do not need to migrate any unmodified example files from WebLogic Commerce Server and Personalization Server. "Unmodified" means you made absolutely no changes to the files, including no name changes. Instead of migrating the files, you can substitute the new versions of those example files by copying them to the appropriate location in your 4.0 directory.

Excluding these files from the migration provides two benefits:

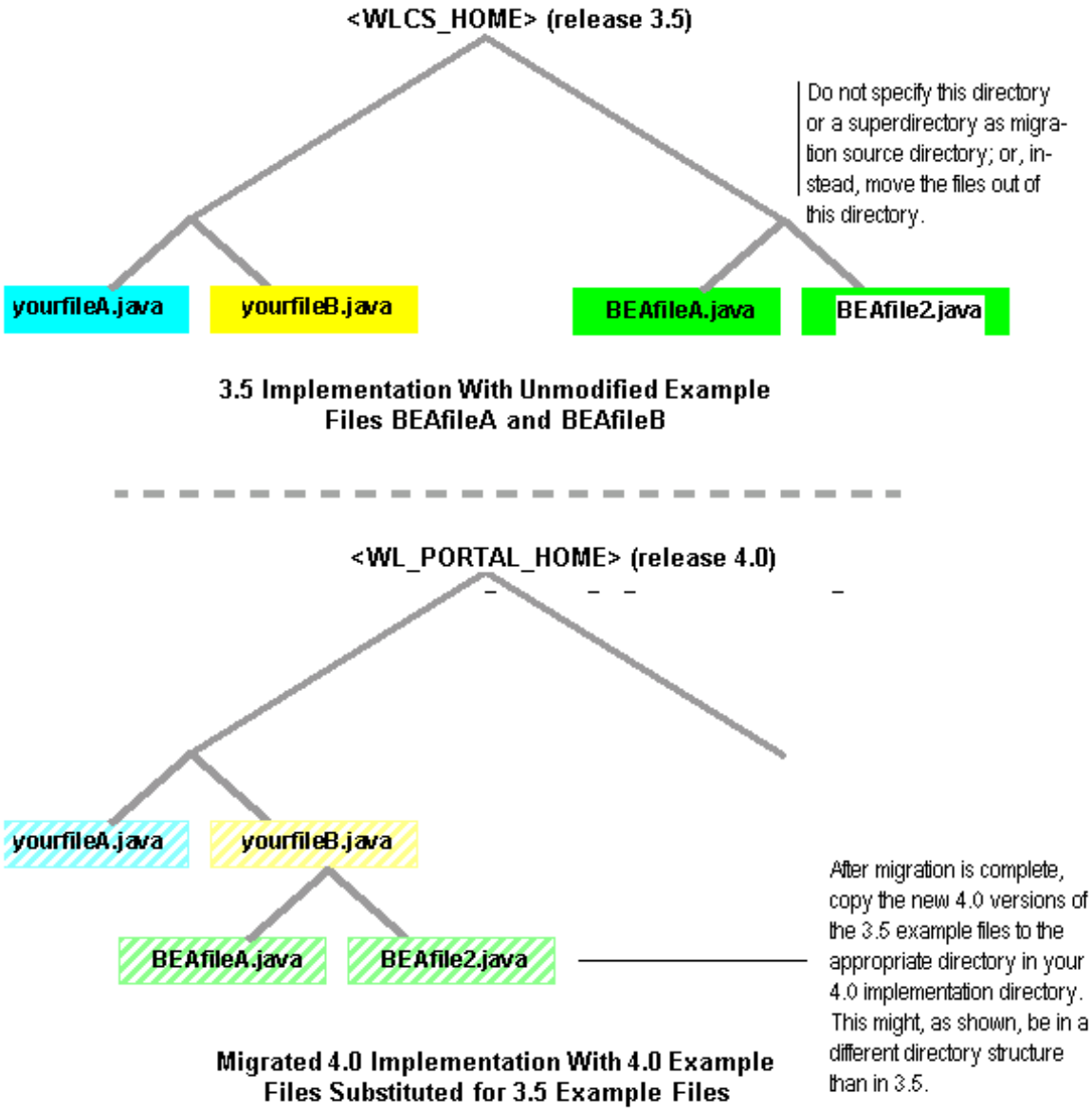
- It saves you the work that would have been involved in migrating them.
- It means that you can be sure the 4.0 example files conform to the new API, if it has changed, as well as new Webflow and other aspects that might have changed from 3.5 to 4.0. Any files you migrate that refer to the unmodified example files will be updated or flagged so that, once you have completed the migration and associated additional coding, they will refer correctly to the new 4.0 versions of the example files.

Follow these steps as you migrate:

- Exclude the unmodified files from the migration. Either move them to a different directory, or do the code migration subdirectory by subdirectory so that you can exclude directories containing unmodified BEA files.
- As you continue through migration, you will come to "Moving New and Migrated Data Files to Your 4.0 Directory" on page 2-56. At that point, copy the 4.0 versions of your unmodified files to the appropriate location in your 4.0 directory.

Figure 2-6 provides an example.

Figure 2-6 Example of Implementation Using Unmodified Example Files

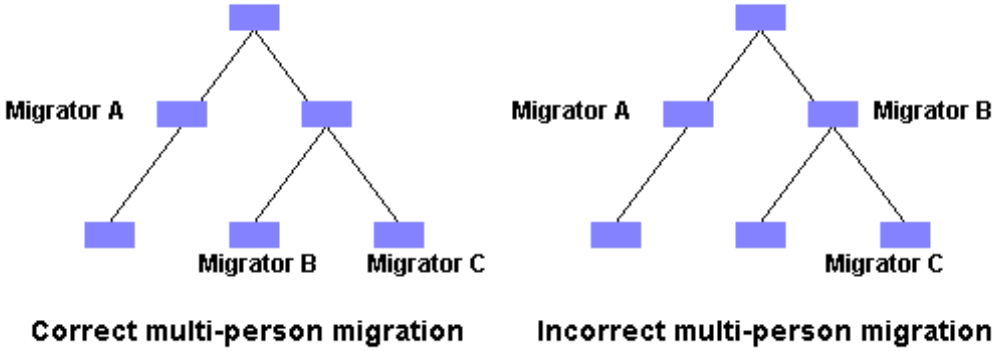


Team Migration Guidelines

To speed up the migration process, multiple people can run the migration simultaneously. For instance, if you use a source code control system, one developer could check out and migrate each of the main subdirectories within your main release 3.5 directory.

To prevent one person's changes overwriting another's, be sure that no one is migrating a subdirectory of code that another person is migrating. This is demonstrated in Figure 2-7.

Figure 2-7 Correct and Incorrect Strategies for Simultaneous Multi-person Code Migration



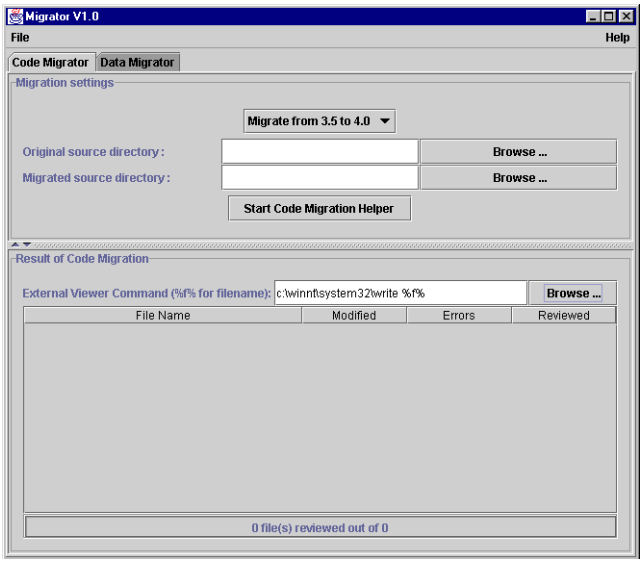
Migrating Code

This section contains the steps for using the utility to migrate your code to 4.0.

For an overview of the process, see "Code Migration Process Overview" on page 2-4.

1. If the migration utility is not already running, start it by double-clicking the `migrator.bat` or `migrator.sh` file at `\migration\bin`.
2. Select the Code Migrator tab, if it is not already on top, as shown in Figure 2-8.

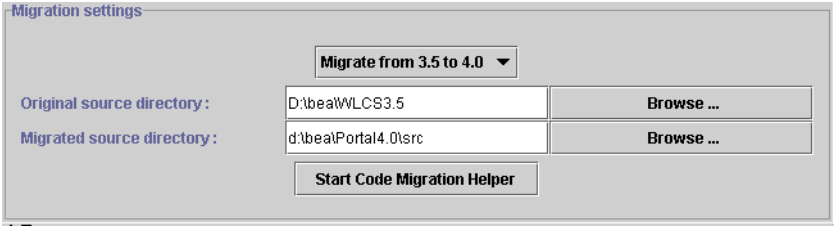
Figure 2-8 Code Migrator Tab



3. Specify a source directory and a destination directory, as shown in Figure 2-9. The source directory is typically the `src` directory in your WLCS 3.5 directory.

Note: The most efficient approach is to migrate the code to a `src` directory within your Portal 4.0 directory. If you do not migrate the data there, you must move it there once the code migration process is over, before you begin reviewing and making changes to your migrated code files.

Figure 2-9 Entering Source and Destination Directories

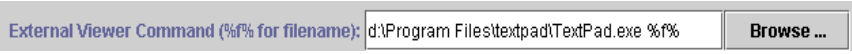


If your implementation contains unmodified BEA example files, follow the guidelines in "Excluding Unmodified BEA Example Files From the Migration Process" on page 2-46.

If more than one person is migrating code, be sure to follow the guidelines in "Team Migration Guidelines" on page 2-48.

4. Enter the path to the executable file for the editor that you want to use to view and edit code files. This is shown in Figure 2-10. (After the migration, you will be able to open any of your code files by double-clicking a file in the list at the bottom of the code migrator window.)

Figure 2-10 Entering Path to External Viewer



Note: You must retain the space and `%f%` at the end of the path; this is the placeholder for the code file that you will be editing.

5. When the migration has run, the migrated files will be listed in the bottom half of the window. You can open the files in your preferred editor by double-clicking them in the window, instead of going to your file system.

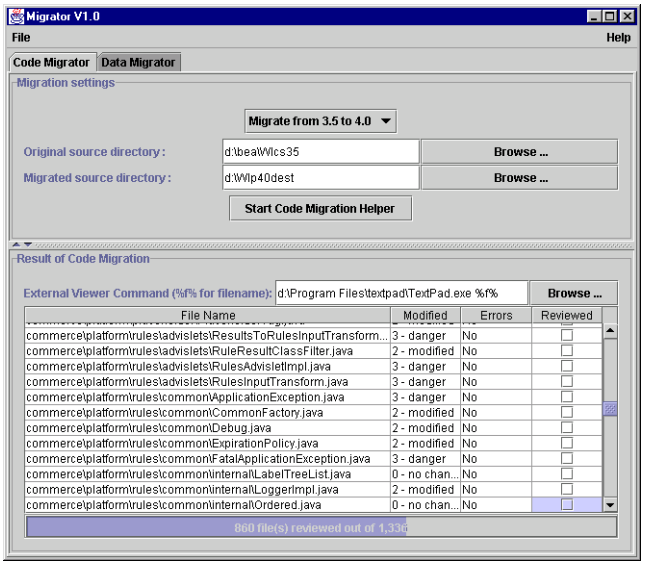
6. The migration process will migrate all the code in the source directory in one process; you do not need to migrate one file at a time. Status messages and the migrated file names will appear during the process.

   When you are ready to migrate the entire source directory, click Start Code Migration Helper.

   If errors occur, refer to the `migration.log` file in the `PORTAL_HOME\migration` directory, the status messages displayed onscreen, and the migration notes added to the migrated code files.

7. When the migration process is complete, the files will be displayed with information about what issues the migration utility encountered, whether the file was changed, and so on. This is shown in Figure 2-11

Figure 2-11 List of Migrated Files and Status



The effect of the migration on each file is indicated by the level 0 through 3, with 3 being most in need of review and additional changes.

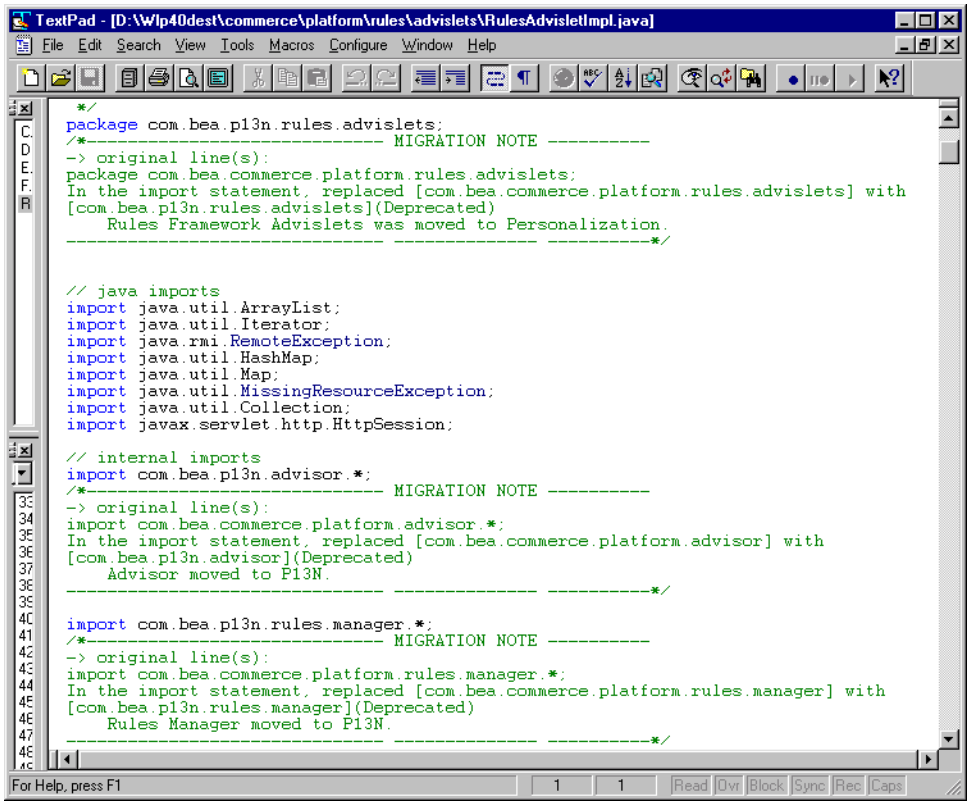
- **0 - no change:** No changes was made to the file. These files are marked as reviewed in the window.
- **1 - comments:** A migration note was added; review the file to determine what if any changes to make.
- **2 - modified:** The file was modified; review the file to determine whether the change was correct, and to view the migration note.
- **3 - danger:** A significant change needs to be made to the file; for instance, it might contain code extending a class that no longer exists.

8. In your destination directory, review each of the migrated files to see what notes were added, and make additional changes to the files according to the migration notes.

Note: The migrated code files need to be in your `PORTAL_HOME\src` directory before you begin making changes.

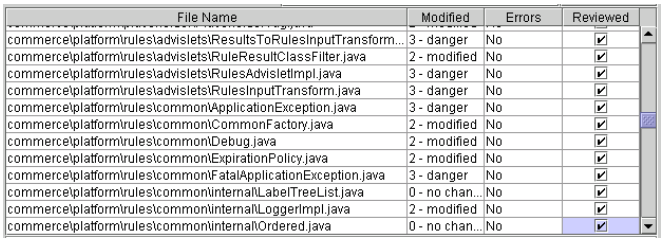
An example of a migrated file and migration notes is shown in Figure 2-12.

Figure 2-12 Migrated Code File



9. Mark the Reviewed box in the migrator utility window after you have completely finished with each file, as shown in Figure 2-13

Figure 2-13 Reviewed Box for Migrated Code Files



File Name	Modified	Errors	Reviewed
commerce\platform\rules\advislets\ResultsToRulesInputTransform...	3 - danger	No	☑
commerce\platform\rules\advislets\RuleResultClassFilter.java	2 - modified	No	☑
commerce\platform\rules\advislets\RulesAdvisletImpl.java	3 - danger	No	☑
commerce\platform\rules\advislets\RulesInputTransform.java	3 - danger	No	☑
commerce\platform\rules\common\ApplicationException.java	3 - danger	No	☑
commerce\platform\rules\common\CommonFactory.java	2 - modified	No	☑
commerce\platform\rules\common\Debug.java	2 - modified	No	☑
commerce\platform\rules\common\ExpirationPolicy.java	2 - modified	No	☑
commerce\platform\rules\common\FatalApplicationException.java	3 - danger	No	☑
commerce\platform\rules\common\internal\LabelTreeList.java	0 - no chan...	No	☑
commerce\platform\rules\common\internal\LoggerImpl.java	2 - modified	No	☑
commerce\platform\rules\common\internal\Ordered.java	0 - no chan...	No	☑

Migrating JSPs

This section covers the changes to JSPs in this release, and what you need to accommodate those changes.

Note: Do not migrate JSPs associated with portals; portals have changed dramatically in this release and it is more efficient to re-create them in 4.0. You will be prompted to do so in "Re-creating Portals and Portlets" on page 2-67.

1. Create `.java` files for each of your JSPs. (For two ways to do this, see "Generating .java Files for JSPs" after these steps.)

   The .java files will be created in a path starting at your WEB-INF directory for the application. For instance, if the JSPs are at `mywebapp\myJSPs\`*`JSPfilenames`*`.JSP`, the generated `.java` files would be created at `WEB-INF\_tmp*\jsp_compiled\_myJSPs\*`*`JSPfilename`*`.java`

2. Complete the steps in the previous section, "Migrating Code" on page 2-48, to run the code migrator on the generated `.java` files. For a destination directory, select a location that is not in your `PORTAL_HOME` directory.

   Before you can begin reviewing and making changes to the JSPs, move the JSPs the appropriate directory in the `PORTAL_HOME` directory. They need to be in the 4.0 environment for you to verify whether the changes were effective. Typically this should be `PORTAL_HOME\applications\wlcsApp-project\`*`webappdir`*`\`.

3. Review each migrated generated `.java` file and the comments and changes in each. The relevant line number in the JSP is referenced in the migration note in the `.java` file, and the corresponding JSP filename and path is added to the top of each `.java` file.

4. Make the necessary changes in your JSPs.

Repeat these steps until no more conversions or warnings show up in the code migrator.

Generating `.java` Files for JSPs

You can do this in a number of ways:

- Add the follow to your `web.xml` file, then open each JSP:

```
<!-- JSP configuration -->
```

```
<jsp-descriptor>
  <jsp-param>
    <param-name>keepgenerated</param-name>
    <param-value>true</param-value>
  </jsp-param>
 </jsp-descriptor>
```

- Quicker: You can use `jspcPrepare.bat` to do this, which is included in the `PORTAL_HOME\migration\bin` directory. The file is shown here. You can either edit the bold parts of the file, then run `jspsPrepare.bat`, or run each command separately from the command line, supplying the appropriate values.

Listing 2-4 Contents of `jspcPrepare.bat`

```
echo off

REM ---------------------------------------------------------#
REM  Helps with JSP files migration by generating
REM  .java files.
REM ---------------------------------------------------------#

SETLOCAL
CALL ..\..\bin\win32\set-environment.bat

REM ---------------------------------------------------------#
REM         VARIABLES TO SET: THIS IS FOR THE WLS 6.0 SP1!!
REM         FOR WLCS 3.5 NOT THE WLS6.1 FOR PORTAL 4.0
REM ---------------------------------------------------------#
set WEBLOGIC_DIR=d:\bea\wlserver6.0    [The path to your 6.0 SP1 installation]
set WLCS_CLASSES=d:\bea\WLCS35\classes [The path to the classes directory
REM                                     for Release 3.5.]

REM ---------------------------------------------------------#
REM         VARIABLES TO SET:
REM ---------------------------------------------------------#
REM
set JDK_1-3_BIN=d:\bea\jdk131\bin  [The path to JDK 1.31 for WLS 6.0 SP1]

REM ---------------------------------------------------------#
REM          The preparation
REM ---------------------------------------------------------#
```

```
set
classpath=%WEBLOGIC_DIR%\lib\weblogic.jar;%WEBLOGIC_DIR%\ext\weblogic-tags.jar;
%WLCS_CLASSES%

REM ------------------------------------------------------#
REM          Show how this should be used
REM ------------------------------------------------------#

REM  [In the path below, enter the path to the files that you want to compile,
REM  instead of <your files>.jsp. You can enter multiple paths, by repeating
REM  the entire line, and you can use * to represent all JSPs in the directory.
REM  The generated .java files will be created at the top of the directory
REM  for the Web application the JSPs belong to.]
echo Before running the migrator (migrator.bat) for JSP files, you need.
echo to compile the JSP pages into .java files. To do so, go to the Web App
directory
echo and run the following command on the JSPs you want analyzed:
echo -
echo "%JDK_1-3_BIN%\java weblogic.jspc -keepgenerated <your files>.jsp"
echo -
echo  You should see all the _yourFile.java files on which you can run the
echo  migrator.


echo on
```

Manual Migration Tasks for 3.5 to 4.0

This section covers how to manually migrate items not migrated by the migration utility. You might not to complete all tasks, depending upon your implementation.

We recommend that you complete the tasks in the order shown.

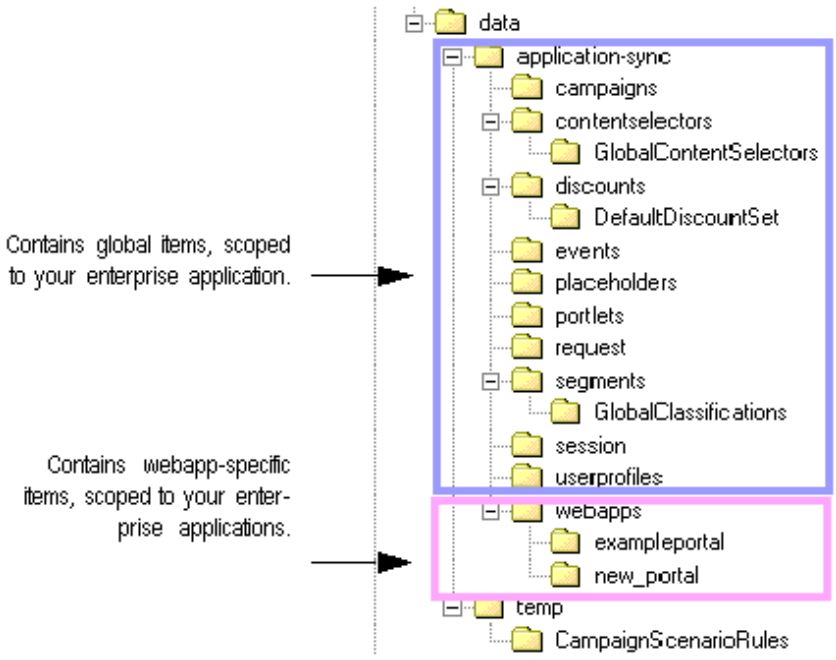
- Moving New and Migrated Data Files to Your 4.0 Directory
- Migrating Deployment Descriptors and Other Configuration Files
- Entering Information Formerly Stored in weblogiccommerce.properties
- Updating Integrations With CyberCash and TAXWARE
- Migrating Webflow and Pipeline

- Migrating non-User and non-Group Configurable Entity Successor Property Values
- Migrating Events
- Re-creating Portals and Portlets
- Checking for Additional Migration Steps

Moving New and Migrated Data Files to Your 4.0 Directory

New XML files and other files were created in your data migration destination directory Figure 2-14 is an example; your directory will vary depending on what features you use and the names you applied to some items.

Figure 2-14 Data Migration Destination Directory



Move all items from the destination directory to the appropriate directory in your `PORTAL_HOME` directory. The complexity of this process will vary, depending upon your implementation. Keep in mind the following guidelines.

Names Might Be Different

Keep in mind that many elements of your implementation will not only be in a different directory structure, but might be renamed. (For instance, see “Naming and Renaming Rules for Placeholders and Other Items” on page 2-91 and “Significant Webflow and Pipeline Changes” on page 2-102.)

Copy Configuration Files to PORTAL_HOME

The configuration files control your system, your enterprise application, or Web application. Copy `config.xml` from your 3.5 system to `PORTAL_HOME\config\`*`domain`*`.` Copy application.xml and application-config.xml

from the `data_dest_dir` to `PORTAL_HOME\applications\`*`domain`*`]META-INF`. Copy your `web.xml` and `weblogic.xml` files to the appropriate stored in each `WEB-INF` in your 4.0 system.

Copy *.wf and *.pln Files to Each Web Application Directory

The information formerly in the `pipeline.properties` and `webflow.properties` was converted and stored in the `data_dest_dir` directory. These files contain all the information for your implementation. In 4.0, you can break this information down according to the namespace it belongs to, so you will need to do editing of this file at a later time. For now, copy the files to each of your Web application directories, in `PORTAL_HOME\applications\wlcsApp-project\application-sync\webapps`. Once migration is complete, follow the instructions in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* to set up your Webflows and Pipelines.

Integrate Files from Repository Directory

The Portal Repository directory is not included in this release. Follow these steps to incorporate any 3.5 files stored in a repository directory. Be sure to perform the steps in the order shown.

1. Copy the contents of the 3.5 repository directory to a different directory, like D:\repository.
2. Copy the contents of your portal Web application into the new repository directory, allowing files with the same name to replace the existing ones.
3. According to your implementation, determine where the contents of the new repository directory should go in your 4.0 implementation

Copy JSPs to Appropriate Web Application Directory

Copy migrated JSPs, if you have not already done so, to `PORTAL_HOME\applications\wlcsApp-project\`*`webappdir`*`\`.

Copy the Entire `application-sync` Directory to `PORTAL_HOME`

An `application-sync` directory was created at `data_dest_dir\data\`. Copy that directory to `PORTAL_HOME\applications\wlcsApp-project` and let it replace the `application-sync` directory that was installed there.

Migration Recognizes Only One Enterprise Application

If you have more than one enterprise application, copy the `data_dest_dir\data\application-sync` directory to `PORTAL_HOME\applications\other_enterprise_app_dir`. Once you start using Portal 4.0, set each one up appropriately if you want any of the enterprise-application-scoped items in that application-sync directory to be different between enterprise applications.

Copy 4.0 Versions of Stock Files at This Time

Also keep in mind that if you followed the instructions in "Excluding Unmodified BEA Example Files From the Migration Process" on page 2-46, you should *copy* the new 4.0 example files to their appropriate spot in your own 4.0 implementation. Example files are documented in the following. Commerce templates are covered in In addition to the templates available in the *Guide to Registering Customers and Managing Customer Services*. For information related to the product catalog, see the *Guide to Building a Product Catalog*. For information on services related to order and purchase processing, see the *Guide to Managing Purchases and Processing Orders*.

Migrating Deployment Descriptors and Other Configuration Files

Deployment descriptors are not migrated by the migration tool. Examine your deployment descriptors and other XML configuration files, keeping in mind any customizations you have made to accommodate additional Enterprise JavaBeans or other items, and update the files as needed.

For more information, refer to the *Deployment Guide*.

Entering Information Formerly Stored in weblogiccommerce.properties

The `weblogiccommerce.properties` file contained several entries that now are stored in two new XML files, `application-config.xml` and `application.xml`. Some of the data was migrated, but you need to verify it and reenter some data at this point, using the WebLogic Server Administration Console. (See the Architectural Overview for more information.) Default values are provided for some properties, which might not be appropriate for your system.

Print or open a copy of your 3.5 `weblogiccommerce.properties` file and follow the instructions in the *Deployment Guide*to ensure that all information is correct.

Warning: Do not directly edit any of the XML files created through the migration; always edit them through the console. It is extremely likely that editing the files directly will cause significant problems for your site.

Updating Integrations With CyberCash and TAXWARE

The previous payment model and tax model were tightly coupled with Cybercash/TAXWARE native libraries. The models provided a payment.tax solution out of the box. The 4.0 implementation introduces a different payment and tax model, based on Web services. This model has a generic payment and tax API, instead of one that is tightly coupled with Cybercash/TAXWARE native libraries. This new generic model allows payment and tax support from other potential service providers.

However, you must replace the sample Web services supplied out of the box with this new model with real, production-grade services.

In addition, evaluation copies of Cybercash and TAXWARE are no longer included with BEA WebLogic Portal.

Refer to “Integrating With a Payment Service” and “Integrating With a Tax Service” in the *Guide to Managing Purchases and Processing Orders* for more information about the APIs in this release.

The original payment EJB (CreditCardService) called native Cybercash libraries. In this release, this EJB invokes a placeholder payment Web service that you must enhance to provide real backend payment functionality. The same requirement applies to tax services.

For information about creating a Web service, see "Programming WebLogic Server Web Services".

All payment and tax properties have been moved to Mbeans in the `application-config.xml` file, and some default properties may have replaced the values in your 3.5 implementation. Use the WebLogic Server Administration Console to set up the correct properties.

Note: Consider changing only the `host:port` portion of the WSDL URL, i.e. `localhost:7501`, for these Web services. Although these properties are now in Mbeans, you should still consider refraining from changing any of these properties without restarting WebLogic Portal. The nature and use of these properties are not intended to be dynamic. They were moved to Mbeans only to facilitate server management and configuration.

Migrating Webflow and Pipeline

This section covers the additional migration necessary for Webflow and Pipeline.

- Setting up Webflow
- Migrating to the New Webflow Editor

Setting up Webflow

The structure for Webflow has undergone significant, complex changes in this release to introduce new features such as allowing multiple namespaces within a Webflow. Refer to the "Webflow and Pipeline" on page 2-101 if you have not read it already for an overview of the changes, and *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* for information on how to write code for a 4.0 implementation.

Check the main.wf and main.pln by opening them in the Webflow Editor and Pipeline Editor, and making any necessary changes.

Migrating to the New Webflow Editor

The old Webflow Editor has been replaced by a new one. To ensure that the new editor is running correctly, follow this checklist.

1. Verify that you have run the Data Migration Webflow and Pipeline tasks. They take the old files and create new files that reflect the new file structure.
2. In your 4.0 directory, run the Webflow Editor and verify that it is functioning correctly. (For more information, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.)
3. Synchronize the file when you are satisfied that the editor is working correctly. (You can do this now or later once all other EBCC data is set up; see "Syncing Data to the Server" on page 2-68.) Once the 4.0 WebApp is ready on the server side through the sync, you are using the 4.0 Webflow Editor and Webflow.

Migrating non-User and non-Group Configurable Entity Successor Property Values

Configurable entities have successors, which allow you to specify what group or user to use in case a customer does not have a group or user of his or her own. WebLogic Portal 4.0 recognizes `Group` and `User` as possible values for successors. If you created any other types of successors, you must migrate them yourself. This section describes the data and code structure of the items you need to modify, and how the migration works for `Group` and `User` successors. Use this information to determine what steps you need to take to migrate your additional successors.

Configurable Entity Successors

In release 3.5, a Configurable Entity could have a property named `_successor\u007e` whose value is a `BLOB`. In release 4.0, the value of a successor property is a `long`, which is the ENTITY_ID of the entity that is the successor. The `BLOB` is a serialized instance of a class called `PersistableHandle`. The `PersistableHandle` is deserialized into a runtime object and contains the `SmartKey` and a `String`, the JNDI home name of the successor. The `SmartKey` is cast to a specific type of entity bean primary key class and an identifier is obtained. The

identifier and the JNDI home name are used to look up the ENTITY_ID of the successor in the ENTITY table. The PROPERTY_VALUE table is then updated, replacing the `BLOB` value of the successor with the `long` value.

The migration tool can only support entities with JNDI home names of `com.beasys.commerce.axiom.contact.Group` and `com.beasys.commerce.axiom.contact.User`. The primary key cannot be known for any other types that you created yourself.

The PROPERTY RDBMS Migration task performs the migration of all other properties and copies the successor property values into the PROPERTY_VALUE_TEMP40 table where this task expects to find them. The _TEMP40 table is used to allow this task to be repeated. Values are obtained from the PROPERTY_VALUE_TEMP40 table and updates are made to the PROPERTY_VALUE table.

Migrating Events

If you are not using events in 3.5, ignore this section.

Note: The events discussed here are the events that are stored in your database. They are *not* the events that you can view in the EBCC that are defined using the property set schema, the `.evt` files. These are fully migrated through the migration utility's data migration process.

Events are actions taken by users, such as clicking on a particular option in your site. You can choose to track events in a database, and analyze them with the tool of your choice, such as Broadbase. (WebLogic Portal does not provide analysis tools for events.)

The way in which events are stored in this release has changed somewhat. The table and columns are the same, but one column containing definition information has changed. This column stores information about the event such as the name of the event and the data. The data in that column is stored, not as a standard string of text, but in XML, so the column data type is a CLOB (character large object).

The way that these categories are stored in the XML file has changed: for instance `event_date` has changed to `event-date`.

You need to complete one or more of the following tasks:

1. Decide whether to migrate any of your events to the 4.0 format, and if so what portion of the data.
2. If you decide to migrate, migrate the appropriate amount of data to the 4.0 format
3. If you decide to migrate, adapt your event data analysis tool, such as Broadbase, to read the new format.
4. Verify that your events that are set up in the EBCC correspond appropriately to your migrated events.

Deciding Whether to Migrate Events

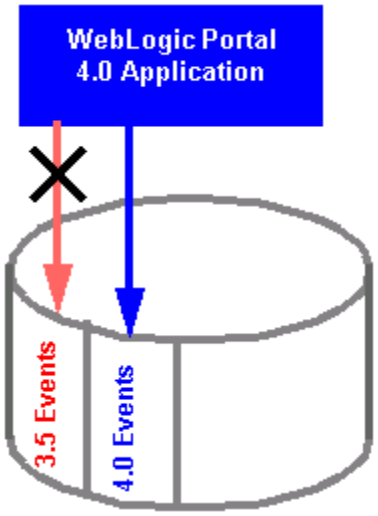
You do not need to migrate your 3.5 events. You need to migrate them only if you want to simultaneously analyze 3.5 and 4.0 events data using the same events analysis tool. For example, if you want to analyze all the events from September through December 2001, analyzing data collected prior to and after migration, they must all be in 4.0 format.

You also must address how much of your events data you want to migrate. If you have been tracking events for some time, the database can become extremely large. If any of your events data is currently in backed-up databases not connected to your system, you must restore that data, run the migration, and presumably return the migrated events data to its original backed-up state. How you manage that process of restoring is up to you.

Migrating Events Data

If you are going to migrate some or all of your 3.5 events data, follow these steps. The steps will create a new empty set of tables, using the 4.0 format, in the database where your 3.5 events are stored. You will then convert the 3.5 events to 4.0 format, and update your event tracking configuration information to log events in the 4.0 tables.

Figure 2-15 Migrating Events Data



1. In your 3.5 events database, create a new database account. It must have a different name than the current account you use to access events.
2. Using that account, run the Behavior Tracking and Events DDL scripts that are used to create the new empty 4.0 events table. See "Persisting Behavior Tracking Data" for instructions. This chapter includes information on the scripts, as well as associated database setup and WebLogic Server Console configuration.
3. In WebLogic Portal, change the connection pool information to point to the new database. Be sure to update the host, port, database, and account information. This will change event logging so that all events are now logged in the new empty 4.0 tables.
4. Update the new events database to comply with the new 4.0 format.

   To transform the data to the new format, you will be using an XSL file, `event.xsl`, that converts the data to the format; and a Java class `EventXSLTest` that runs the XSL template. (The fully qualified name of the class is `com.bea.commerce.migration.data.version.v3_5to4_0.EventXSLTest`.)

   The following code shows an example of how to create a `.bat` or `.sh` file using the XSL file and the Java class. The exact entries are up to you to determine,

based on the variables in your system; the lines you need to set are in bold text. The event.xsl file was installed on your system within the PORTAL_HOME\src\commerce\migration\data\version\v3_5to4_0\xsl\ directory.

Listing 2-5 Example .bat or .sh file to convert events CLOB to new 4.0 format

```
SETLOCAL

CALL ..\..\bin\win32\set-environment.bat

set JDK_1-3_BIN=%BEA_HOME%\jdk131\bin

set MIGRATION_DIR=%WL_COMMERCE_HOME%\migration

set MIGRATION_LIB=%MIGRATION_DIR%\lib

set
MIG_CLASSPATH=%BEA_HOME%\lib\tools.jar;%MIGRATION_LIB%\migration.jar;%MIGRATION
_LIB%\apache\xerces-1_3_1\xerces.jar;%MIGRATION_LIB%\apache\xalan-j_2_0_1\xalan
.jar;%BEA_HOME%\wlserver6.1\lib\weblogic.jar;%BEA_HOME%

echo on

set XSL_FILE=[path_to_event.xsl_file]\event.xsl

%JDK_1-3_BIN%\java -cp %MIG_CLASSPATH%  -DMIGRATION_DIR=%migration_dir%
com.bea.commerce.migration.data.version.v3_5to4_0.EventXSLTest %XML_FILE%
%XSL_FILE%
```

When you have created the file, run it from the command line or by double-clicking it.

5. If you have backed-up databases that you also need to migrate, restore them one by one and repeat all the steps in this section.

Migrating Your Event Data Analysis Tool

If you want to use your event data analysis tool with 4.0 events, you must update it to comply with all the XSD files in the PORTAL_HOME\migrationlib\schema\4.0 directory of the 4.0 installation. If you have not migrated all your events to 4.0 format, keep a version of your analysis tool that works with the 3.5 event data format.

Compare EBCC .evt Property Set Files to Migrated Events

Ensure that your .evt files are set up appropriately in the EBCC for the way you are using events. See the *Guide to Using the E-Business Control Center* for more information.

Re-creating Portals and Portlets

Portlets were partially migrated. Only a template file was created for each portal, with minimal information, because of the significant changes and new features in this release. The portal file name is an encoded version of the value of the `NAME` column of the WLCS_PORTAL_DEFINITION and the file has a `.portal` extension. The file for each portal is created in
`<miration_dest_dir>/data/application-sync/webapps/<portal_name>/`

The creation of the portal report is described in "Webflow Migration" on page 2-17, the migration of the portals and portlets themselves is described in "Portal / Portlet Migration" on page 2-35.

1. Open the portal report file, created at
   `migration_dest_dir\data\PortalReport.txt`
2. Refer to the instructions in the *Getting Started with Portals and Portlets* to use the new BEA Portal 4.0 EBCC, and choose new skins, layouts, set up groups, and complete any additional tasks for each of your portals.

Listing 2-6 shows an example migrated portal file. This is provided only to give you an idea of what data is migrated; do not edit any of your portal files directly.

Listing 2-6 Example Migrated Portal XML File

```
<?xml version="1.0" ?>
<result-set>
  <row>
    <column name="NAME" value="Defined Portlets" />
    <column name="CONTENT_URL" value="portlets/definedportlets.jsp" />
    <column name="HEADER_URL" value="##NULL##" />
    <column name="ALTERNATE_HEADER_URL" value="##NULL##" />
    <column name="FOOTER_URL" value="##NULL##" />
    <column name="ALTERNATE_FOOTER_URL" value="##NULL##" />
```

```
    <column name="TITLEBAR_URL" value="titlebar.jsp" />
    <column name="BANNER_URL" value="##NULL##" />
    <column name="EDITABLE" value="0" />
    <column name="EDIT_URL" value="##NULL##" />
    <column name="HELP" value="0" />
    <column name="HELP_URL" value="##NULL##" />
    <column name="ICON_URL" value="##NULL##" />
    <column name="MINIMIZEABLE" value="1" />
    <column name="MAXIMIZEABLE" value="1" />
    <column name="MAXIMIZED_URL" value="##NULL##" />
    <column name="MANDATORY" value="0" />
    <column name="MOVEABLE" value="1" />
    <column name="FLOATABLE" value="1" />
    <column name="MINIMIZED" value="0" />
    <column name="LOGIN_REQUIRED" value="0" />
  </row>
… any number of more rows…
</result-set>
```

Checking for Additional Migration Steps

Review “Migration Reference Information” on page 2-71 again and make any additional changes at this time that are necessary for your implementation. This could include changes based on the information in “User Group Hierarchy” on page 2-93, “Caching” on page 2-100, “Cybercash and TAXWARE” on page 2-92, “LDAP Entity Property Manager” on page 2-98, and so on. Each section contains references to where you can find the information you need in the BEA documentation set.

Syncing Data to the Server

In this release, you set up items like campaigns and portlets using the E-Business Control Center, then implement them by using the data sync feature. When you are ready to lynch EBCC data to the server, refer to the *Guide to Using the E-Business Control Center* to complete this task.

Verifying the Migration From 3.5 to 4.0

The migration is a complex task, which might involve hundreds or thousands of separate files, depending on your implementation. We strongly recommend that you thoroughly verify the migrated 4.0 version of your software using the topics in this section, and any other steps you deem appropriate.

General Verification

In a test environment, run a typical selection of common user and administrator tasks to be sure your system was migrated correctly. Suggested tasks include the following:

- Go to the example portal and be sure you can log in. See the Architectural Overview.
- Go to the example commerce services application and make sure you can both view items and add them to a shopping cart.
- Log into the EBCC and view properties for several users. See the *Guide to Using the E-Business Control Center* for more information.
- Refer to the items you noted as significantly affecting your implementation, based on the information in "Migration Reference Information" on page 2-71. Check the functionality of each item on the list.

Refer to the appropriate section of the referenced documentation and this guide if you encounter errors.

EBCC Data Verification

To verify that all your EBCC data from the previous version was migrated correctly, follow these steps:

1. Launch the EBCC without the WebLogic Portal server running. (See *Guide to Using the E-Business Control Center* for more information.)
2. In the menu, choose File > Open Application.

3. In the Select an Application Directory window. locate and single-click the application folder. For example, to open the sample application, single-click the `sample_app` folder.
4. Click Open. Date for the application is loaded into the E-Business Control Center. When an application is loaded, you can select a tool icon in the Explorer window and see its files.

Final Verification

The key indicators of whether all aspects of the migration have fully succeeded are whether the following tasks run correctly:

- Synchronizing data from the EBCC to the server
- A representative array of standard customer and administrative tasks

We strongly recommend that you run exhaustive real-world tests like these, and others appropriate for your business, before considering the migration complete.

What's Next: Getting Started With BEA WebLogic Portal 4.0

Now that you have successfully completed migration, use the following information to get started using BEA WebLogic Portal 4.0.

Licenses

There is a license conversion when you migrate from WebLogic Server 6.0 to 6.1; if you will be doing this at the same time you migrate from WebLogic Personalization Server 3.5 to 4.0, see "Upgrading Licenses from WebLogic Server 6.0" at `http://e-docs.bea.com/wls/docs61/install/instlic.html#1036261`

Getting Started

Use the online documentation site to locate the documentation you need for additional learning.

`http://edocs.bea.com/wlp/docs40/index.html`

You might also want to review the following guides:

- *Strategies for Developing E-Business Web Sites*
- *Deployment Guide*
- *Guide to Creating Portals and Portlets*

Migration Reference Information

Use the following throughout the migration to enhance your knowledge of 4.0 changes.

This section covers the notable changes to this release. Be sure to review all of them, and make any changes necessary, based on how the changes affect your implementation.

For an overview of the features in the new migration tool, see "Getting Started With the Migration Tool" on page 2-2. Be sure to also read the "What's New" page at `http://e-docs.bea.com/wlp/docs40/interm/whatsnew.htm` for a preview of the new features in this release.

The information is organized as follows:

- How to Use This Information
- Enhancements in This Release
- Database Changes
- Code Changes
- JSP Tags

- New Client and Server Directory Structures
- Data Setup and Deployment Through EBCC
- Cybercash and TAXWARE
- Behavior Tracking
- Effect on LDAP Configuration
- User Group Hierarchy
- Changes to User Management
- Changes to weblogiccommerce.properties File
- Data Setup and Deployment Through EBCC
- Caching
- Webflow and Pipeline
- Events
- Content Management Link on Tools Administration Site

How to Use This Information

This section covers, at a high level, the changes in this release. You need to read this section to be aware of all the changes that have occurred, so that you can implement new features, understand the extent of changes for features you have implemented, and so on.

As you review this section, make a list of items that affect your implementation. You will need to be aware of them when you review each code file during code migration.

Once you have finished the migration process, you will be directed to check your list in “Checking for Additional Migration Steps” on page 2-68. At that point, if you have not already made the necessary changes through the migration utility, subsequent reviews and changes, and the manual migration steps later, make the necessary changes. Each change in this section contains a reference to the information you will need.

You can also use the list you create as a tool for verification, in "Verifying the Migration From 3.5 to 4.0" on page 2-69.

Enhancements in This Release

The numerous changes in this release make the migration somewhat more time consuming. However, the changes are necessary to provide significant enhancements, including the following:

- Flow Manager has been replaced with Webflow throughout this release
- Multiple namespaces within a Webflow
- Data synchronization
- Portal layouts that include portal pages, skins, and multi-portal architecture
- Improved performance
- Improved maintainability
- Easier integration with third-party vendors
- Ease of use

Database Changes

Finding Information on 4.0 Schemas

The Database Schemas at `http://edocs.bea.com/wlp/docs40/interm/schemas.htm` is a central point from which you can easily find any of the 4.0 schemas you want to look at.

Each 4.0 guide on a feature that uses databases also has a schemas chapter:

- *Guide to Developing Campaign Infrastructure*
- *Guide to Building a Product Catalog*
- *Guide to Events and Behavior Tracking*

- *Guide to Managing Purchases and Processing Orders*
- *Getting Started with Portals and Portlets*
- *Guide to Building Personalized Applications*

Transition From Databases to XML File Data Storage

Much of the data that was stored in standard databases in the previous release will be stored in individual XML files after you migrate. You will not access and edit the data directly in the XML files, but through the E-Business Control Center.

Warning: Do not directly edit any of the XML files created through the migration; always edit them through the EBCC. The XML files are extremely sensitive and editing them directly could cause significant problems for your site.

We recommend that you store the files in a source control system, and make frequent backups.

The subsequent sections provide more information about what data is affected and where it is now stored.

Selected Descriptions of Data Converted to XML

The following items were stored in databases in 3.5, and are stored in XML files in 4.0; additional changes took place, where noted. See also "Naming and Renaming Rules for Placeholders and Other Items" on page 2-91 for information on changes to the items' names.

- Placeholders – Stored in one file per placeholder; the filename is the placeholder name.
- Portals and portlets – Each portal and portlet is stored in a separate file; the filename is the portal or portlet name.
- Rules – Each rule that is not tied to a specific campaign is stored in a separate file; the file is based on the rule name.
- Discounts – Each rule is stored in a separate file; the file is based on the rule name.

- Campaigns – Campaigns and scenarios are migrated, and scenarios are embedded in their associated campaigns. Any rules dependent on scenarios are then brought into the migrated campaigns, as well.

  Each campaign is stored in a separate file; the file is based on the campaign name. Included in the campaign file are the associated rules and scenarios.

3.5 and 4.0 Data Storage

The subsequent tables list the tables for the previous release and where that data is stored in this release.

New Requirement for DBLoader and Product/Category Database Records

When you use DBLoader to add products to your catalog, you must ensure that for every record in WLCS_PRODUCT and WLCS_CATEGORY, there is a corresponding record in the CATALOG_ENTITY table. This was not a requirement in release 3.5, but is a requirement for release 4.0.

For more information about using DBLoader, see "Using the Product Catalog Database Loader" in the *Guide to Building a Product Catalog*.

New Tables in 4.0

New tables (no data to migrate):

- DATA_SYNC_APPLICATION
- DATA_SYNC_ITEM
- DATA_SYNC_SCHEMA_URI
- DATA_SYNC_VERSION

3.5 Tables and Corresponding 4.0 Tables or Files

4.0 data converted directly from a specified 3.5 table are shown in Table 2-7

Table 2-7 3.5 Tables Converted or Carried Over to 4.0

3.5 Table	Corresponding 4.0 Table or File
WLCS_IS_ALIVE	WEBLOGIC_IS_ALIVE

Table 2-7 3.5 Tables Converted or Carried Over to 4.0

3.5 Table	Corresponding 4.0 Table or File
WLCS_SEQUENCER	SEQUENCER (Consolidates information from the 3.5 table WLCS_UID, as well. WLCS_UIDS is used by Portal only in release 3.5, and is not used in 4.0.)
WLCS_SCHEMA	XML based
WLCS_PROP_MD	XML based
WLCS_PROP_MD_BOOLEAN	XML based
WLCS_PROP_MD_DATETIME	XML based
WLCS_PROP_MD_FLOAT	XML based
WLCS_PROP_MD_INTEGER	XML based
WLCS_PROP_MD_TEXT	XML based
WLCS_PROP_MD_USER_DEFINED	XML based
WLCS_ENTITY_ID	ENTITY
WLCS_PROP_ID	PROPERTY_KEY
WLCS_PROP_BOOLEAN	PROPERTY_VALUE
WLCS_PROP_DATETIME	PROPERTY_VALUE
WLCS_PROP_FLOAT	PROPERTY_VALUE
WLCS_PROP_INTEGER	PROPERTY_VALUE
WLCS_PROP_TEXT	PROPERTY_VALUE
WLCS_PROP_USER_DEFINED	PROPERTY_VALUE
WLCS_USER	USER_SECURITY
WLCS_GROUP	GROUP_SECURITY
WLCS_USER_GROUP_CACHE	USER_GROUP_CACHE
WLCS_GROUP_HIERARCHY	GROUP_HIERARCHY

Table 2-7 3.5 Tables Converted or Carried Over to 4.0

3.5 Table	Corresponding 4.0 Table or File
WLCS_USER_GROUP_HIERARCHY	USER_GROUP_HIERARCHY
WLCS_LDAP_CONFIG	No longer used
WLCS_UNIFIED_PROFILE_TYPE	No longer used
WLCS_UUP_EXAMPLE	No longer used
CAMPAIGN	XML based
SCENARIO	XML based
CAMPAIGN_SCENARIO	XML based
SCENARIO_CLASSIFICATIONS	XML based
SCENARIO_END_STATE	SCENARIO_END_STATE
AD_BUCKET	AD_BUCKET
AD_COUNT	AD_COUNT
MAIL_ADDRESS	MAIL_ADDRESS
MAIL_BATCH	MAIL_BATCH
MAIL_BATCH_ENTRY	MAIL_BATCH_ENTRY
MAIL_HEADER	MAIL_HEADER
MAIL_MESSAGE	MAIL_MESSAGE
PLACEHOLDER	XML based
PLACEHOLDER_PREVIEW	PLACEHOLDER_PREVIEW
WLCS_DOCUMENT	DOCUMENT
WLCS_DOCUMENT_METADATA	DOCUMENT_METADATA
WLCS_UIDS	Consolidated with WLCS_SEQUENCER into SEQUENCER. No longer used.

3.5 Portal Tables and 4.0 Portal Tables and Files

3.5 Portal tables that are no longer used are shown in the following list:

- WLCS_PORTAL_DEFINITION
- WLCS_PORTLET_DEFINITION
- WLCS_COLUMN_INFORMATION
- WLCS_PORTAL_PERSONALIZATION
- WLCS_GROUP_PERSONALIZATION
- WLCS_USER_PERSONALIZATION
- WLCS_PORTAL_GROUP_HIERARCHY
- WLCS_PORTAL_HIERARCHY
- WLCS_CATEGORIES
- WLCS_TODO
- WLCS_BOOKMARKS
- WLCS_RULESET_DEFINITION
- RULESET
- WLCS_CAT_ENTITY_ID
- WLCS_CAT_PROP_ID
- WLCS_CAT_PROP_BOOLEAN
- WLCS_CAT_PROP_DATETIME
- WLCS_CAT_PROP_FLOAT
- WLCS_CAT_PROP_INTEGER
- WLCS_CAT_PROP_TEXT
- WLCS_CAT_PROP_USER_DEFINED
- WLCS_CATEGORY

- WLCS_PRODUCT
- WLCS_PRODUCT_CATEGORY
- WLCS_PRODUCT_KEYWORD
- WLCS_CURRENCY
- WLCS_COUNTRY
- WLCS_CUSTOMER
- WLCS_SHIPPING_ADDRESS
- WLCS_CREDIT_CARD
- WLCS_TRANSACTION
- WLCS_TRANSACTION_ENTRY
- WLCS_SAVED_ITEM_LIST
- WLCS_ORDER
- WLCS_ORDER_LINE
- WLCS_SHIPPING_METHOD
- WLCS_SECURITY
- DISCOUNT
- DISCOUNT_ASSOCIATION
- DISCOUNT_SET
- ORDER_ADJUSTMENT
- ORDER_LINE_ADJUSTMENT

New 4.0 tables are shown in the following list:

- PORTAL
- PORTLET
- LAYOUT
- SKIN

- PORTAL_PAGE
- PORTAL_PAGE_P13N
- PORTAL_PAGE_P13N_LAYOUT
- PORTAL_P13N
- PORTAL_P13N_ADMIN
- PORTAL_P13N_SKIN_POOL
- PORTLET_PLACEHOLDER
- PORTLET_P13N
- USER_PORTAL_PAGE_P13N
- USER_PORTAL_P13N
- USER_PORTLET_P13N
- WLCS_TODO
- WLCS_BOOKMARKS
- CATALOG_ENTITY
- CATALOG_PROPERTY_KEY
- CATALOG_PROPERTY_VALUE
- CATALOG_PROPERTY_VALUE
- CATALOG_PROPERTY_VALUE
- CATALOG_PROPERTY_VALUE
- CATALOG_PROPERTY_VALUE
- CATALOG_PROPERTY_VALUE
- WLCS_CATEGORY
- WLCS_PRODUCT
- WLCS_PRODUCT_CATEGORY
- WLCS_PRODUCT_KEYWORD

- WLCS_CUSTOMER
- WLCS_SHIPPING_ADDRESS
- WLCS_CREDIT_CARD
- WLCS_TRANSACTION
- WLCS_TRANSACTION_ENTRY
- WLCS_SAVED_ITEM_LIST
- WLCS_ORDER
- WLCS_ORDER_LINE
- WLCS_SHIPPING_METHOD
- WLCS_SECURITY
- DISCOUNT
- DISCOUNT_ASSOCIATION
- ORDER_ADJUSTMENT
- ORDER_LINE_ADJUSTMENT

Data Setup and Deployment Through EBCC

The structure and processes for EBCC use and data deployment have changed significantly. The following sections outline key changes; please see the *Guide to Using the E-Business Control Center* for more information.

Coordinating Client and Server Data

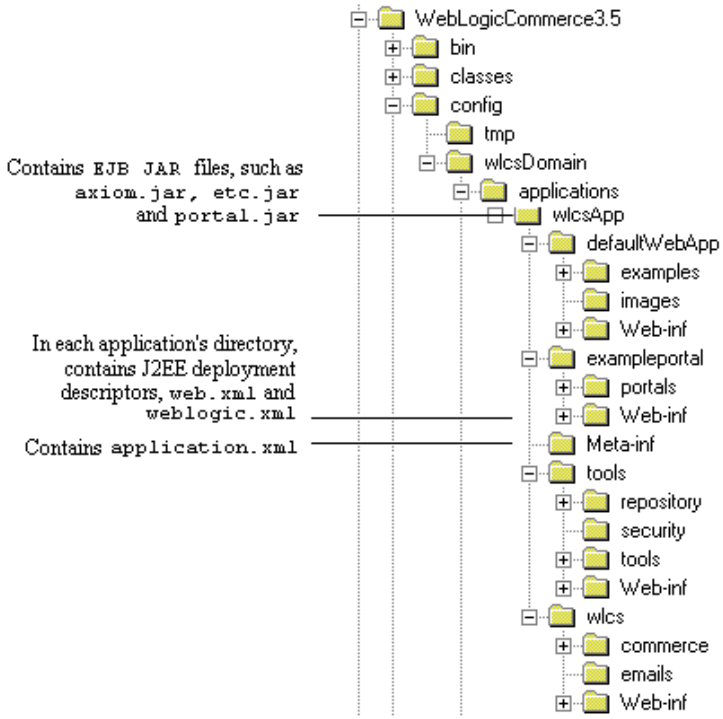
A new data deployment mechanism is provided in this release. The data that you manage through the E-Business Control Center, such as information about placeholders and campaigns, is now managed (viewed and edited) locally on a client, rather than on the same computer as the server. To make apply your changes to an application, you must complete a data synchronization process, sending the data to the server.

It also means that you must include the DataSync Web application in your enterprise applications. Without it, you cannot deploy campaigns or similar items to your Web applications.

New Client and Server Directory Structures

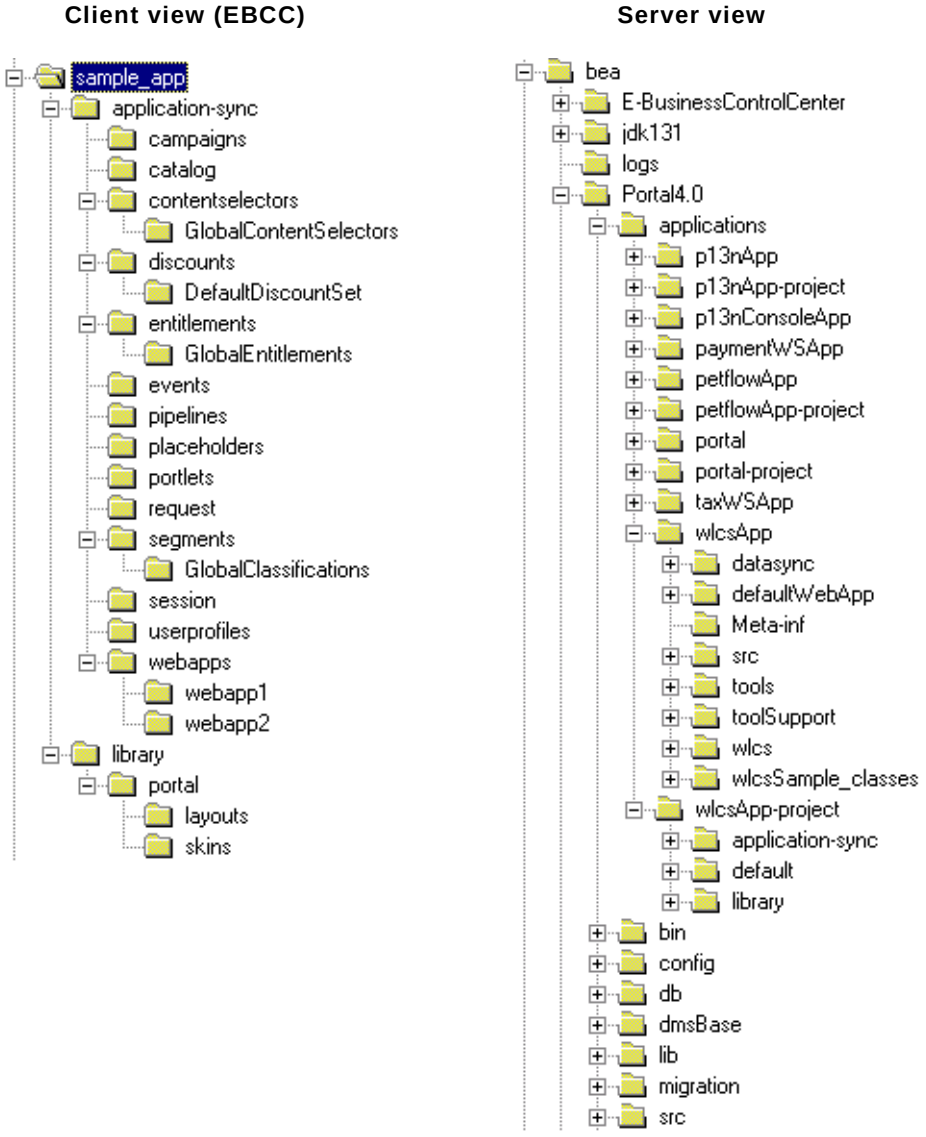
Figure 2-16 shows a 3.5 directory structure; Figure 2-17 shows the structure for 4.0. After code and data migration, you will be instructed to move the files to the appropriate locations in your implementation's 4.0 directory structure.

Figure 2-16 WebLogic Commerce Server 3.5 sample directory structure



The directory structure in this release is different; there is a client directory structure and a server directory structure.

Figure 2-17 Sample directory structure for a valid 4.0 application



Deploying Rule Sets

Rules sets are represented as XML files that you can view and edit using the E-Business Control Center. In order to make your application use the rule set changes you have made in the EBCC, you need to complete a procedure that synchronizes the rule set XML documents with the appropriate target application (the application you want affected by the rule set changes). The procedure uses the Personalization Server's Data Synchronization Framework (for more information, see the *Deployment Guide*).

When you perform this synchronization process, the following occur:

- The previously mentioned Data Synchronization Framework *propagates* the changed rule set XML documents to an instance of the Rules Manager, in the target application.
- The Rules Manager parses the new rule set XML document and changes it to a new binary representation of the data. This binary representation is the format that the BEA Rules Engine can read.
- From that point on, the target application makes will use the new rule sets when it makes calls to the Rules Manager, while executing rules.

Note: The rule set XML documents, which are managed by a particular instance of the Rules Manager, are scoped to that application. That is, changes to rule sets are seen only by the target application, and not by any other application. It is also important to note that changes to rule sets are immediate, and, unlike previous releases' functionality, the Rules Manager no longer relies upon a time-to-live in order to propagate rule set changes throughout a cluster.

Code Changes

The changes to code in this release are described in the `migrinfo.html` file, included in the `PORTAL_HOME\migration\doc` directory. For more information, see "Reference Information Files for Code Changes in This Release" on page 2-10 in the "Getting Started With the Migration Tool" section, and the 4.0 JavaDocs.

Note: Some items have been removed, rather than being deprecated. The `migrinfo.html` file states whether this has occurred.

Note: To search and sort the information, use the Migration Viewer tool. See "Viewing Code Changes Using the Migration Viewer Tool" on page 2-105 for more information.

JSP Tags

You can easily locate all information on 4.0 JSP tags from this location:

`http://edocs.bea.com/wlp/docs40/interm/jsptags.htm`

The following information highlights key changes, in the following sections:

- Location of .tld Files for Example Templates
- JSP Tag <ph:placeholder> and name Attribute
- General Changed and New JSP Tags
- Webflow Tags
- Personalization Tags

Location of .tld Files for Example Templates

The provided `.tld` files are now located at
`PORTAL_HOME\applications\wlcsApp\wlcs\WEB-INF`

JSP Tag <ph:placeholder> and name Attribute

The `name` attribute for `<ph:placeholder>` now needs to be the URI of the deployed placeholder document.

Example of old format:

```
<ph:placeholder name="Main Page Banner"/>
```

`Main Page Banner` came from the `<name></name>`attribute in the placeholder XML file of old format.

Example of new format:

```
<ph:placeholder name="/placeholders/Main_Page_Banner.pla"/>
```

`/placeholders/Main_Page_Banner.pla` is the URI of the data deployed placeholder.

General Changed and New JSP Tags

- Changed Tags
- Deprecated Tags
- Removed Tags

Changed Tags

`getPipelineProperty` is now `getProperty`.

Deprecated Tags

`es:simpleReport`

Removed Tags

The following tags have been removed; suggestions for what to use instead, where applicable, are provided.

`es:preparedStatement` – The tag has no replacement or similar functionality in this release.

`um:changeGroupName` – The tag has no replacement or similar functionality in this release.

`um:getChildGroups` – `um:getChildGroupNames` has the same functionality.

Webflow Tags

The JSP tags for Webflow and Pipeline have changed drastically. See *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* [for more information.

Personalization Tags

- **<pz:> Personalization Tags** – No changes.

- **<fm:> Flow Manager** – All Flow Manager tags have been ***removed*** from the Personalization Server. The following provides information about how to complete the tags' tasks in different ways.
  - `<fm:getApplicationURI>` You can get the request URI from the HTTP session.
  - `<fm:getCachedAttribute>` Use pipeline session to store and retrieve attributes.
  - `<fm:getSessionAttribute>` Use pipeline session to store and retrieve attributes.
  - `<fm:removeCachedAttribute>` Use pipeline session to store and retrieve attributes.
  - `<fm:removeSessionAttribute>` Use pipeline session to store and retrieve attributes.
  - `<fm:setCachedAttribute>` Use pipeline session to store and retrieve attributes.
  - `<fm:setSessionAttribute>` Use pipeline session to store and retrieve attributes.
- **<ps:> Property Sets**
  - `<ps:getRestrictedPropertyValues>` – ***New tag.*** Returns a list of restricted values for a specific property definition, converted into Strings.
  - `<ps:getPropertyName>` – The attribute `schemaGroupName` has been changed to `propertySetType`.
  - `<ps:getPropertySetNames>` – The attribute `schemaGroupName` has been changed to `propertySetType`.
- **<um:> User Management**
  - `<um:getChildGroups>` – ***Tag removed.*** Use `<um:getChildGroupNames>` instead.
  - Wherever `UserManagerTagConstants` appears, it is now `UserManagementTagConstants`. For example:

    ```
    <%@ page import="com.beasys.commerce.user.jsp.tags.
    UserManagerTagConstants" %>
    ```

    is now:

```
<%@ page import="com.bea.p13n.usermgmt.servlets.jsp.taglib.
UserManagementTagConstants" %>
```

- `<um:getProfile>` – Users are no longer configurable entities. The retrieved profile can be treated as a `com.bea.p13n.usermgmt.profile.ProfileWrapper`. The attributes `successorType` and `profileType` were removed; they are not used anymore.
- For all of these Group-User Management tags:

```
<um:addGroupToGroup>
<um:addUserToGroup>
<um:changeGroupName>
<um:createGroup>
<um:createUser>
<um:removeGroup>
<um:removeUser>
<um:removeUserFromGroup>
```

the documentation previously posted a note which read:

Note: This tag should only be invoked when the class `com.beasys.commerce.axiom.contact.security.RDBMSRealm` is defined as the active security realm. This can be verified in the WebLogic Server Administration Console.

The note has been changed to read:

Note: This tag should only be invoked when the current realm is an implementation of `weblogic.security.acl.ManageableRealm`. This interface is implemented by the default WebLogic Personalization Server realm (`com.bea.p13n.security.realm.RDBMSRealm`).

- `<um:getTopLevelGroups>` – Retrieves an array of group names, each of which has no parent group. The information is taken from the realm. (It is no longer taken from the personalization database tables.)
- `<um:getUsernames>` – Can now be used with any realm.
- `<um:getUsernamesForGroup>` – Can now be used with any realm.
- <um:setPassword> – This tag should only be invoked when the current realm is an implementation of `weblogic.security.acl.ManageableRealm`. This interface is

implemented by the default WebLogic Personalization Server realm (`com.bea.p13n.security.realm.RDBMSRealm`).

In addition, the user object used by the current realm must implement `weblogic.security.acl.CredentialChanger`.

- **<cm:> Content Management**
  - <cm:select> – Has a new attribute:`contextParams`.
  - <cm:selectById> – Has two new attributes: `onNotFound` and `contextParams`.
- **<ads:> Ads** – No change.
- **<ph:> Placeholders**
  - `<ph:placeholder>` – The `name` attribute for now needs to be the URI of the deployed placeholder document. See "JSP Tag <ph:placeholder> and name Attribute" on page 2-86.
- **<i18n:> Internationalization** – No change.
- **<es:> Personalization Server Utilities**
  - <es:simpleReport> deprecated.
  - <es:monitorSession> removed
  - <es:convertSpecialChars> – ***New tag***.
- **<wl:> WebLogic Server Utilities** – No change.

Example JSP Templates

The appearance and functionality of the templates included in the product have not changed significantly.

- Events in the templates now use a new notation: instead of `link(logout)`, for example, the event is now written as `link.logout`.
- The templates are divided into multiple namespaces. The namespaces used are: `sampleapp_main`, `sampleapp_search`, `sampleapp_order`, and `sampleapp_user`.

The templates are documented in the Guide to Registering Customers and Managing Customer Services, as well as in the Webflow diagrams at http://edocs.bea.com/wlcs/docs35/interm/webflo.htm.

Application Scoping

Application scoping is new in this release; most configuration data is applicable to the entire enterprise application.

Naming and Renaming Rules for Placeholders and Other Items

Names of your rules, discounts, portals, portlets, and placeholders might change because of naming restrictions in this release. In previous releases, you could apply any name to a rule, campaign, or placeholder. In 4.0, however, names must conform to the following rules:

- File names must be at least one character and up to 64 characters (60 character name, a dot separator and a 3-character extension).
- The first character must be a letter, ideograph (a symbol such as &, $, or @), or underscore.
- Any other characters can be a letter, ideograph, digit, underscore, hyphen, or dot (period).

Any name not conforming to those restrictions will be converted during the data migration by the migration utility using the following formula.

- If the first character is not a letter or underscore, an underscore will be added to the beginning of the name.
- In the rest of the name, any character other than those allowed will be replaced by an underscore.
- If the name is longer than 60 characters (excluding a 3-character extension), the name is truncated at the end to 60 characters.

Cybercash and TAXWARE

Cybercash and TAXWARE are no longer used with this release. The tax and payment APIs have been updated to allow you to connect to different tax and payment services.

All payment and tax properties have been moved to Mbeans in the `application-config.xml` file. You can modify it using the WLS Administration Console.

- Taxes – The Tax MBean is stored in the <TaxServiceClient> element of the `application-config.xml` file. The "TaxCalculatorWSDL" attribute identifies the tax Web service WSDL URL that tells WebLogic Portal where the tax Web service can be found. (The `application-config.xml` file is located at `<PORTAL_HOME>\applications\wlcsApp\META-INF`.)
- Payments – In the same `application-config.xml` file, payments are controlled in the same way though the <PaymentServiceClient> element and "PaymentWebServiceWSDL" attribute.

You will be prompted to update any relevant code in your implementation, in “Updating Integrations With CyberCash and TAXWARE” on page 2-60.

Refer to “Integrating With a Payment Service” and “Integrating With a Tax Service” in *Guide to Managing Purchases and Processing Orders* for more information.

Behavior Tracking

The following changes were made to behavior tracking functionality for this release:

- The Pipeline Component `ShoppingCartTrackerPC` changed to conform to new Pipeline Component classes.
- The `webflow.properties` and `pipeline.properties` files were migrated to tracking-scoped XML files, `main.pln` and `main.wf`.

User Group Hierarchy

Significant changes were made in this release to the User Group hierarchy. This affects the personalization services and portals, since they share the hierarchy.

- Time Lag in Entitlements Updating
- The User Group Hierarchy Changes
- Notes on Managing the User Group Hierarchy Changes
- Effect on LDAP Configuration

Time Lag in Entitlements Updating

A user's association to a list of entitlement segments is cached and updated every hour, or earlier if you restart the server. If you change a user's profile property values to alter his or her segment association—for instance, upgrading a user from Developer to Experienced Java Developer—the changes will not appear until an hour has passed or you have restarted the server.

This happens only when you change a change a user's profile property values, or any other activity that qualifies him or her for a different number of entitlement segments.

If you instead use the "add/remove entitlement segments" link, this actually creates *entitlement definitions* for that resource, for that segment, for that user. (All three are needed for an entitlement definition). Entitlement definitions are not cached; only the association of a user with all segments that he or she qualifies for (sometimes called the "Roles"). If you add/remove entitlement segments, you are actually adding/removing entitlement definitions. Doing this does not require waiting an hour or rebooting the server.

The User Group Hierarchy Changes

Key changes include the following; a diagram of the new hierarchy is shown in Figure 2-18 later in this section.

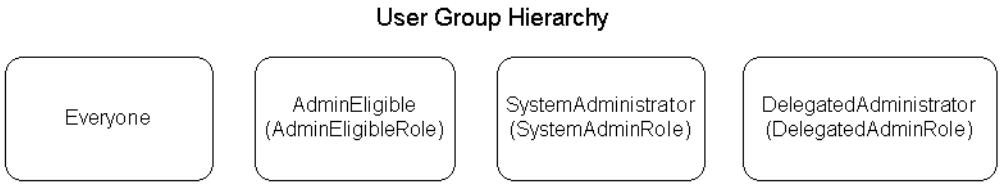
- The existing `admin` user group has been renamed; it is now called `SystemAdministrator`.

- The `AdminRole` J2EE security role has been renamed; it is now called `SystemAdminRole`.
- A new user group was added: `DelegatedAdministrator`. The `DelegatedAdministrator` has a subset of the `SystemAdministrator` tasks.

  Note: `DelegatedAdministrator` is supported *only* in WebLogic Portal; `SystemAdministrator` is supported by all products.

Regarding anonymous users, this release uses the term "everyone" to refer to users who do not belong to a user group or have not logged in yet. This is because they are implicitly members of the `Everyone` group.

Figure 2-18 Release 4.0 User Group Hierarchy



- Everyone – Reserved WebLogic Server User Group. Every User is automatically associated with this User Group. This User Group should not be added manually to the SecurityRealm because it is already available from WebLogic Server.
- AdminEligible (AdminEligibleRole) – Allows differentiation between visitor users and admin-eligible users to prevent accidents.
- SystemAdministrator (SystemAdminRole) – Replaces existing admin group and J2EE AdminRole.

  System Administrators have full control over all WebLogic Portal products and are the only administrators who can access the system tools.
- DelegatedAdministrator (DelegatedAdminRole) – Any delegated administrators (administrators without full SystemAdmin privileges) are associated with this user group and role. Specific privileges to resources are maintained by the Delegated Admin tables and the Entitlements Service.

Refer to *Getting Started with Portals and Portlets* for more information, and to the *Security Guide* for release 3.5 to compare this information with how user groups worked in the previous release.

Notes on Managing the User Group Hierarchy Changes

A significant number of `.java`, `.jsp`, and XML files are affected by this change; the migration tool will attempt to migrate or flag all changes, but you are as always strongly encouraged to review all affected files yourself. You will need to make sure that the changes are configured correctly in your underlying concrete realm implementation.

If after migration, strange behavior occurs in your site when you log in as an admin, it is likely that all necessary changes regarding these user group updates have not been made. Review the migration log file and all migration notes in migrated code.

Effect on LDAP Configuration

The new administrative hierarchy means you need to change your LDAP configuration. See the *Guide to Building Personalized Applications*, and the topic "Creating and Managing Users for more information.

Changes to User Management

The Commerce services use `User` and `Group` profile data to provide personalized content and behavior. This data is stored and accessed through the `User` and `Group` profile managers, which use implementations of the `EntityPropertyManager`. Additionally, a Unified Profile Type framework is provided to allow application developers to create custom `EntityPropertyManager` implementations and have transparent access to properties that are stored in other data stores, such as a legacy system.

Change to UserManagement Components User, Group, and UserManager

Any services that use the `UserManagement` components (`User`, `Group`, and `UserManager`) must now use the new `UserManagement` components (`com.bea.p13n.usermgmt` and `com.bea.p13n.profile` instead of `com.beasys.commerce.axiom.contact`).

Separation of Authentication and Authorization

In previous releases, the user/group profile information and user/group security information were combined in the database. In this release, they are separated to better support third-party realm implementation. The design in this release also replaces entity beans with stateless session beans.

See the *Guide to Creating and Managing Users* for more information about user management in this release.

CachedProfileBean Has Been Removed

`CachedProfileBean` has been removed in this release. ProfileWrapper is used instead, using either the new session attribute name to retrieve it, or using the ProfileFactory to create a new one.

User/Group Profile Managers

The `UserProfileManager` and `GroupProfileManager` stateless session beans provide a `ConfigurableEntity`-like interface without the need for instantiating an entity bean. This provides significantly improved performance. `UserProfileManager` and `GroupProfileManager` are responsible for setting and retrieving property values, following the successor hierarchy when searching for properties, and mapping properties to various datasources (accessed by `EntityPropertyManager` implementations).

Support for different user profile types is achieved by re-deploying the `UserProfileManager` with different property-to-datasource mappings. The `UserManager` session bean is responsible for retrieving the correct `UserProfileManager`, given a username.

User/Group Manager Session Beans

The `User` and `Group` Manager session beans provides a way to create, access, and delete `User` and `Group` profiles, and manipulate `User`/`Group` relationships. Within the scope of `User`/`Group` Profiles, it also handles the logic of creating users of a specific profile type, and making sure that when those users are retrieved and deleted, they continue to be treated as that type so that everything will stay synchronized.

These session beans are also meant to be used whenever programmatic access to the realm is required. When a user or group is created through the session bean, it will create both a realm entry and a profile.

Realm Integration

WebLogic Server is currently using the concept of a Realm to provide authentication and authorization. When a user is authenticated anywhere in the system, the request eventually makes its way to an implementation of the WebLogic Server realm interface. WebLogic Server provides a `BasicRealm`, which allows a client to simply access user and group records by name:

- A `ListableRealm`, which allows a client to retrieve lists of users and groups
- A `ManageableRealm`, which allows a client to create new users and groups.

The default Personalization Server realm, an adaptation of the `RDBMSRealm`, is an implementation of `ManageableRealm`.

WebLogic Personalization Server will take advantage of the supplied `RDBMSRealmMBean` to provide database configuration. When this Mbean is active, corresponding configuration screens will be shown in the WLS console. These screens are shipped out-of-the-box with WLS. The WebLogic Personalization Server realm will not use the “schema” property of the Mbean; it will instead externalize SQL statements in a properties file.

Unified Profile Types

Unified Profile Types provide a mechanism for using different types of user profiles within an application. Each different type of user profile might get its properties from different datasources. For example, the `UserProfileManager` session bean may be deployed once in the default manner, with no property mappings; and again as a `CustomerProfileManager`, with mappings set up so that any properties in the Customer property set will be handled by a `Customer` implementation of `EntityPropertyManager`.

When a user is created through the `UserManager` session bean, a profile type can be provided to associate with the user. Later, when the user profile is retrieved through the `UserManager`, it will ensure that the correct profile manager is used based on the user’s profile type.

LDAP Entity Property Manager

The LDAP Entity Property Manager is an implementation of the Entity Property Manager interface that retrieves profile data from an LDAP server. The LDAP Entity Property Manager will get its LDAP connection settings from its deployment descriptor so that it will be de-coupled from the realm. In previous releases, the LDAP Entity Property Manager used the LDAP realm settings. This change will allow properties to be read from multiple LDAP servers, or from an LDAP server when the LDAP realm is not being used.

To use the `LDAPEntityPropertyManager`, a mapping must be created in the UserProfileManager's and GroupProfileManager's deployment descriptor so that certain properties, or entire property sets, will be handled by the `LDAPEntityPropertyManager`.

Realm Configuration Session Bean

The Realm Configuration session bean acts as a synchronization point between the WebLogic realm and the WebLogic Personalization Server database. When a user profile is requested, and it is not found in the database but the username does exist in the realm, a new profile will be created for that user (see section 6.1.1 for details). However, the realm may change, and these users may be deleted. Because WebLogic Server authenticates against the realm, this does not pose a security risk, but the database should be cleaned up periodically. This session bean provides reporting capabilities to show how the database is out of sync with the realm, as well as clean-up functionality to bring things back into order.

User/Group Profile Tags

The User Management tag library includes tags to manage users and groups, as well as tags to access user and group profiles. The user and group profile tags will be addressed here. In previous releases, these included a tag (`getProfile`) that placed a lightweight object (`CachedProfileTag`) representing a profile in the session; and after this was called, other tags could be used to access this profile (`getProperty`, `setProperty`, etc). For this release, it is the responsibility of a Webflow component to place a `ProfileWrapper` object in the session with a well-known variable name (defined in a constants file). When this component is called for the first time, if no user is currently authenticated, it will create an empty `ProfileWrapper`, which will default

to a lightweight "anonymous" profile. When a customer logs in, this event will be caught and a new `ProfileWrapper` will be created with that customer's profile, and will replace the anonymous profile in the session.

The `ProfileWrapper` object itself is a façade to hide either a lightweight anonymous profile, or some combination of `User` and `Group` `ProfileManagers`, depending on how it is initialized. Its purpose is to give access to `User` and `Group` profile data in a way that can be easily modified if performance is poor, and in a way that can be identified by the rules engine.

This is an improvement on the former implementation, because it gives better performance and is simpler.

How Changes Are Migrated

The `User` and `Group` tables will be updated by adding new unique IDs. Code that uses the components described in this section will be migrated to point to the new package structures. The JSP tag interfaces remain the same.

Any code that directly uses a `CachedProfileBean` will be updated to use a `ProfileWrapper`, and to either use the new session attribute name (`BEA_personalization.ProfileWrapper`) to retrieve it, or use the `ProfileFactory` to create a new one. The session attribute is defined as a constant in `com.bea.p13n.usermgmt.SessionHelper.PROFILE_WRAPPER`. There are also methods in `SessionHelper` to retrieve a profile from the session so you do not have to do the lookup yourself: `SessionHelper.getProfile( HttpRequest request )` and `SessionHelper.getProfile(HttpSession session )`.

Code that uses a `User` or `Group` entity bean will be updated to use the `ProfileFactory` and `ProfileWrapper` objects. Code that uses the `UserManager` for group manipulation will use the new `GroupManager` instead.

HttpSession Timeouts and PipelineSession Input Processor

Release 3.5 of Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, provided a way to detect when the `HttpSession` timed out while trying to get the `PipelineSession` Input Processor. Release 4.0 of WebLogic Portal does not provide this functionality.

To address this issue, add the following scriptlet to your JSPs:

```
HttpSession session = request.getSession(false);
if (session.isNew())
        throw new InvalidSessionStateException(..);
```

Changes to weblogiccommerce.properties File

Almost all properties from `weblogiccommerce.properties` have been moved to the following files:

- `application-config.xml`, which contains properties that declare and configure MBeans
- `application.xml`, which contains standard J2EE properties. The migration tool moves data to the appropriate XML files

In this release, instead of editing the files directly, you will use the WebLogic Server Administration Console to modify properties stored there.

Warning: Do not directly edit any of the XML files created through the migration; always edit them through the console. The XML files are extremely sensitive and it is extremely likely that editing them directly will cause significant problems for your site.

Some properties are still stored in `weblogiccommerce.properties`; see "Viewing and Modifying Properties in `weblogiccommerce.properties`" in the *Deployment Guide* for more information. You will be instructed later in this process to verify and if necessary reenter information stored there. These instructions are in "Entering Information Formerly Stored in weblogiccommerce.properties" on page 2-60.

Caching

The entries in `weblogiccommerce.properties` for caching have moved to `application-config.xml`. The entries are migrated during the data migration process.

Key changes to the caching process are listed here; see the *Performance Tuning Guide* for more information.

- Caching is now using MBeans, enabling configuration of the cache from the WebLogic Server console. MBeans also enable cluster-wide notifications on the Cache objects for flushing or invalidating a particular entry in the cache, as well as changing the behavior of the cache at runtime.
- Several methods in the Cache have been removed, simplifying the API to more closely represent the behavior of a cache. This means that some methods that were available in previous releases, and which were marked with the `@deprecated` tag, are no longer available for use in development.
- CacheMBean and the configuration of a Cache no longer has the attribution for listening for notifications. Now all caches are flushable if you know the name. The WLS Administration Console has been updated to reflect the change in the listens attribute.
- CacheFactory API has changed slightly; the method that included the `listens` attribute is gone.
- The mechanism for flushing a cache cluster-wide has changed; it is now `CacheManager.flushAll(name,key)`.

Webflow and Pipeline

Webflow and Pipeline architecture have changed drastically in this release. Refer to the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* for more information on how Webflow works in this release.

You will also need to evaluate your Webflow once you have completed using the migration tool, to make additional changes to take advantage of the new architecture and features. (See “Migrating Webflow and Pipeline” on page 2-61.)

Pipeline has been removed; all functions are now in Webflow.

New Webflow Features

Webflow in this release has the following features:

- Multiple namespaces within a Webflow

- All Webflow and Pipeline configuration is now accomplished using the Webflow and Pipeline Editors, which generate XML files where the data is stored.
- Easier maintenance through centralized configuration
- Decreased interdependence between developers, with the addition of namespaces.

Significant Webflow and Pipeline Changes

Changes include the following.

- The `webflow.properties` and `pipeline.properties` files have been replaced with one or more `<namespace>.wf` and `<namespace>.pln` files. The default is `main.wf` and `main.pln`. Many .wf files may make up a Webflow because of this new namespaces feature. This new implementation is designed to increase the flexibility of the Webflow mechanism.
- You can use namespaces to separate a Webflow into a number of smaller, more manageable modules. Namespaces allow multiple developers to work with separate portions of a Webflow (and thus with separate configuration files and Pipeline Processor session properties) that feed into the larger Webflow for a Web application, without having to worry about naming collisions. For example, a Pipeline Component defined in one namespace can access a variable defined in another namespace, then redirect to a JSP defined in yet a third namespace.
- Several Web application context parameters (defined in your `web.xml` file) have been changed, including the following:
  - WLCS_DEFAULT_NAMESPACE
  - WLCS_APPLICATION_URL
  - WLCS_URL_PREFIX
  - WLCS_STATIC_ROOT
- The Pipeline custom tag library, `pipeline.tld`, is no longer used. All functionality is now in `webflow.tld`.
- There are no longer any Pipeline JSP tags; they are all designated as Webflow tags, since pipelines are a component concept. However, for the most part Pipeline session properties (attributes) did not change.

- `webflowJSPhelper` has been changed to use JSP tags. The `JSPhelper` class still exists, but should be used sparingly because it has been deprecated.
- The Pipeline session is now stored in `HttpRequest` instead of `HttpSession`. This has been changed to enhance performance and facilitate clustering.
- Pipeline exceptions have changed; `PipelineFatalException` and `PipelineNonFatalException`. no longer exist. Exception fatality is now defined within the Webflow `.xml` files.
- The default behavior for the way Webflow generates URLs has changed.
  - The old behavior – If you didn't specify a value for "httpInd" in the Webflow tags, the default value was"calculate".
  - The new behavior – If you do not specify a value for "httpInd" in the Webflow tags, the default value is"http"

  The following tags now have different default behavior:

```
<webflow:createWebflowURL/>
<webflow:form/>
webflow:validatedForm/>a
```

```
<portal:createWebflowURL/>
<portal:form/>
<portal:validatedForm/>a
```

```
<portal:createWebflowURL/>
<portlet:form/>
<portlet:validatedForm/>
```

  This change will boost performance; mostly at startup as calculated values are cached.

Webflow and Pipeline Editor

The new version of the Webflow and Pipeline Editor is called Webflow Editor. The old version is not included in this release. See “Migrating to the New Webflow Editor” on page 2-62 for more information.

Events

If you are using events, you might need to migrate them manually. See “Migrating Events” on page 2-63 for more information about migrating events stored in the database.

Content Management Link on Tools Administration Site

The Content Management link on the main tools administration site will not be displayed in this release if there are no Content Management links in the `weblogiccommerce.properties` file.

WebLogic Portal 4.0

The portal features in release 4.0 are significantly different and new documentation exists for them. See *Guide to Creating Portals and Portlets* for more information.

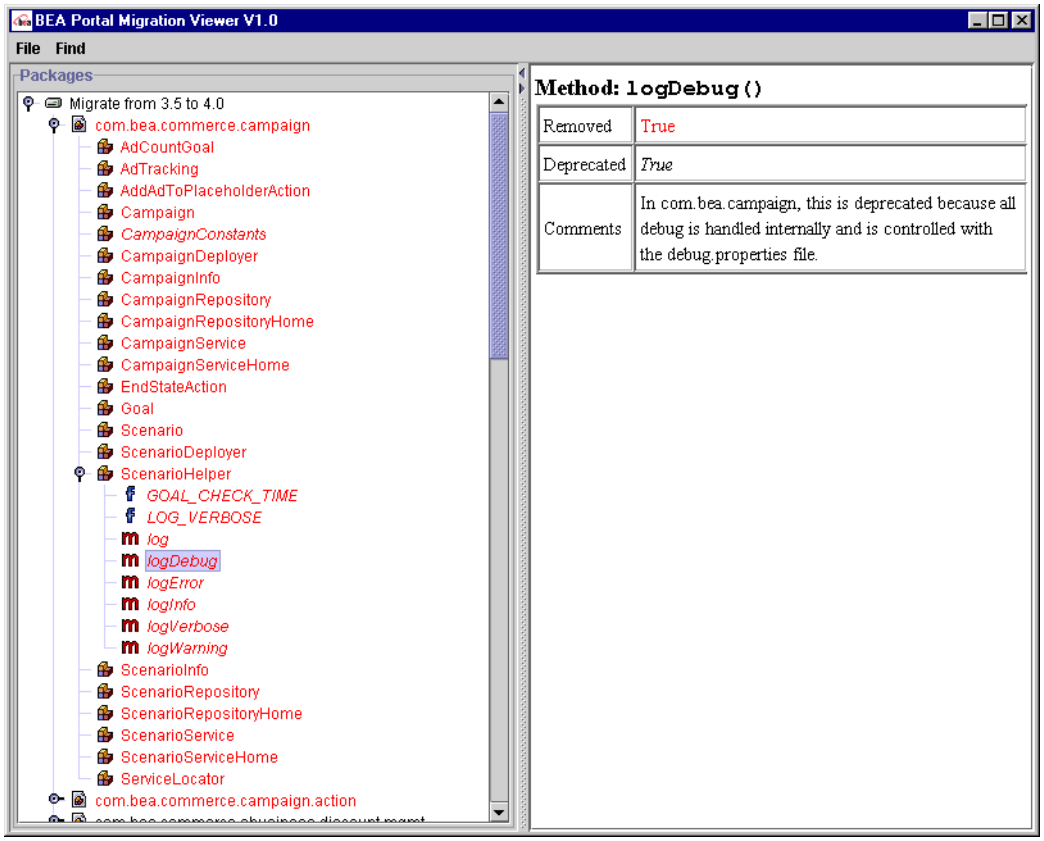
Message Catalog Framework

BEA Portal 4.0 has adopted the new message catalog framework from WebLogic Server. Please see `http://e-docs.bea.com/wls/docs61/i18n/index.html` for complete documentation and usage guides. We strongly recommend that you adopt this framework.

Viewing Code Changes Using the Migration Viewer Tool

The Migration Viewer tool, available on the BEA Developer Center Web site, provides quick, easy access to information about API changes in WebLogic Portal 4.0. The Viewer is shown in Figure 2-19

Figure 2-19 Migration Viewer Tool



This enables you to see more easily at a glance how many of the changes affect your implementation, and how much work will be involved in your migration efforts.

This information was provided in an HTML file (`migrinfo.html`) in release 4.0; the Migration Viewer allows you to find information more easily in a GUI environment. The tool is installed with Service Pack 1 and is also available on the BEA Developer Center Web site (`http://developer.bea.com`).

The Migration Viewer includes the following key features:

- A GUI environment for easier scanning of changes
- Visual cues about types of changes (italics for deprecation, red for removal)
- Icons indicating packages, classes, fields, and methods
- A find feature allowing you to search for a keyword in packages, classes, fields, and methods; in names and in comments.

Installation Instructions

Note: The Migration Viewer requires the `migration/lib/migration.jar` file and a JAXP1.1-compliant XML parser in the CLASSPATH. The `viewer.bat` and `viewer.sh` scripts automatically set this up, if installed into a standard BEA WebLogic Portal 4.0 installation.

The Migration Viewer tool is available at the BEA Developer Center, at `http://developer.bea.com`

Unzip the `migration_viewer.zip` file to your PORTAL_HOME root directory (for instance, `D:\bea\wlportal4.0, or /opt/bea/wlportal4.0`). Be sure to maintain directory names.

The zip file contains the following files:

- `PORTAL_HOME/migration/lib/migration_viewer.jar`: the required class files and images
- `PORTAL_HOME/migration/bin/viewer.bat`: a Windows batch script for starting the viewer
- `PORTAL_HOME/migration/bin/viewer.sh`: a UNIX sh script for starting the viewer

- `PORTAL_HOME/migration/migration_viewer.txt`: a README containing information about the tool, as well as installing and running it.

Starting the Migration Viewer

Locate the `PORTAL_HOME\migration\bin\viewer.bat` (Windows) or `PORTAL_HOME/migration/bin/viewer.sh` (UNIX) file. Run the file to start the Migration Viewer.

Using the Migration Viewer

You can perform two main types of tasks in the viewer:

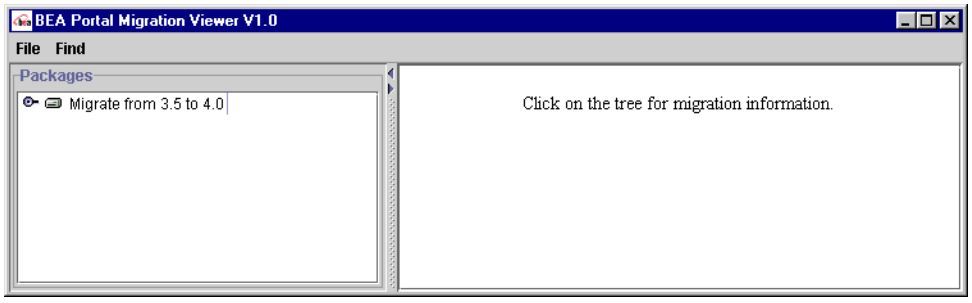
- Use the main GUI window to view information about the listed packages, classes, fields, and methods
- Use the Find window to search for specific keywords

Viewing Information in the Main Window

1. If you have not done so already, start the Migration Viewer.

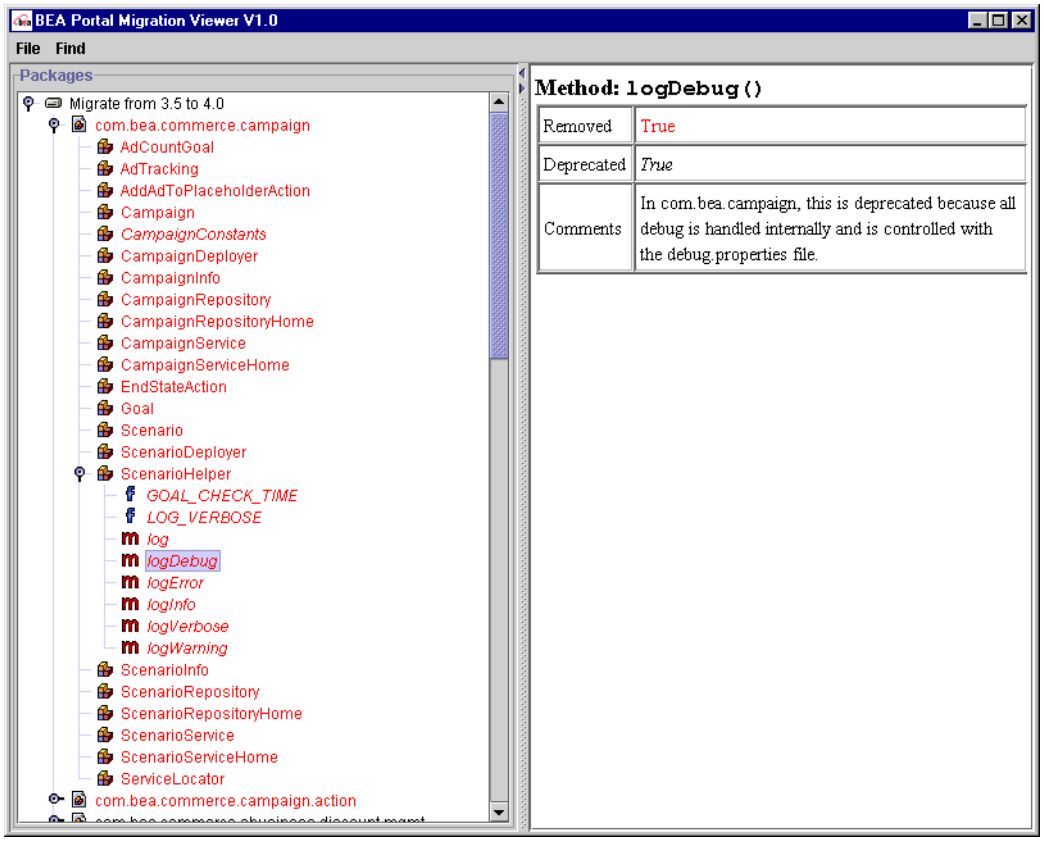
   The Viewer should appear as shown in Figure 2-20:

Figure 2-20 Migration Viewer Tool — Initial View



2. Click in the left panel on the key icon next to “Migrate from 3.5 to 4.0”. The packages for this migration will appear in the left panel, and information about the selected item will appear in the right panel.
3. Expand the icons and select the packages, classes, fields, and methods you want information about, as shown in Figure 2-21.

Figure 2-21 Migration Viewer Tool: Subsequent View After Selecting



Legend for cues in the left panel:

- Red text indicates a removed item.
- Italicized text indicates a deprecated item.

- A key icon indicates a package.
- A boxes icon indicates a class.
- An **M** icon indicates a method.
- An **f** icon indicates a field.

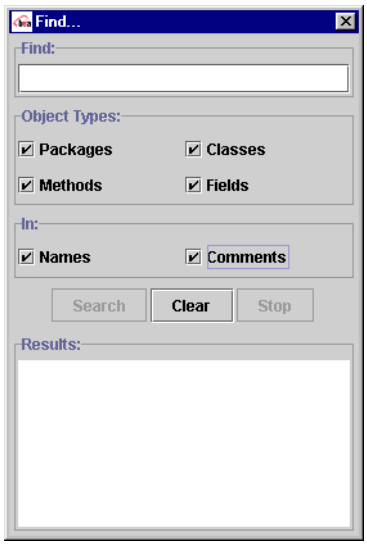
Searching by Keyword in the Find Window

If you want to find a particular keyword anywhere in the API change descriptions, use the Find function.

1. From the Migration Viewer Find menu, choose Find.

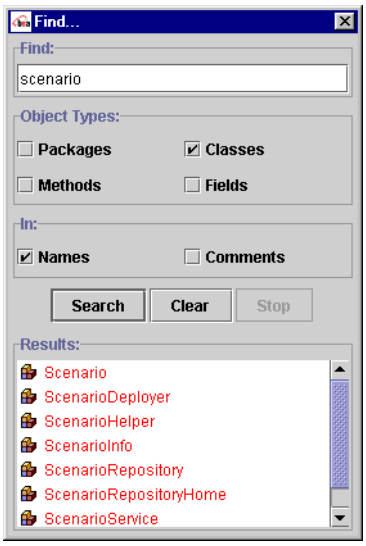
   The Find window appears, as shown in Figure 2-22.

Figure 2-22 Migration Viewer Tool: Finding, Step 1



2. Type the keyword in the Find field.
3. Unmark any types or locations where you do not want to search for the keyword.
4. Click Search; the results will appear in the Results window, as shown in Listing 2-23.

Figure 2-23 Migration Viewer Tool: Finding, Step 2



5. Double-click any item in the Results window; the item will be displayed in the main Migration Viewer window.

CHAPTER

3 Migrating WebLogic Commerce Server From Version 3.2 to 3.5

The WebLogic Personalization Server is bundled with the Commerce Server. This document uses “Commerce Server” to refer to both servers.

Be sure to read the “What’s New” page on the e-docs.bea.com Web site for a preview of the new features in this release:
http://e-docs.bea.com/wlcs/docs35/interm/whatsnew.htm

This chapter addresses the changes to Commerce Server and WebLogic Personalization Server since the 3.2 release.

This section includes the following topics:

- Migration Checklist for 3.2 to 3.5
- What’s New in 3.5
- Migrating Code from 3.2 to 3.5
- Migrating Data from 3.2 to 3.5
- Migrating Your WLCS License
- Verifying the Migration to 3.5
- What’s Next: Getting Started With 3.5

Migration Checklist for 3.2 to 3.5

Read this information thoroughly before continuing; the more closely you follow the information in this section, the more likely your migration will be to go smoothly.

Important Migration Tips and Strategies

The bulk of the code migration consists of your reviewing the new and changed features, locating where in your code you use or depend on any of them, and making the appropriate changes. Before you begin, be sure that you have addressed the following issues, revealed through migration testing.

- Some constants have been changed/removed, and some methods have been removed – Many elements of this release have been removed or changed, rather than deprecated. Be sure that you've thoroughly reviewed your entire system to be sure that all changes and removals described in this document are addressed.
- Pricing Service, formerly known as Discount Manager, is required by the stock Webflow included in this release. For information about using and implementing Pricing Service, see the following documentation:
  - http://e-docs.bea.com/wlcs/docs35/order/discount.htm#1006517
  - http://e-docs.bea.com/wlcs/docs35/campdev/overview.htm#1056731
  - http://e-docs.bea.com/wlcs/docs35/order/shopcart.htm#1046732
  - http://e-docs.bea.com/wlcs/docs35/pdf/tuning.pdf
- If part of your site includes customized BEA stock templates that were not renamed before being put into use, you might encounter challenges during the migration. If you have done so, do the following before you begin:
  - Copy the example "webflow set" into the project source tree.
  - Maintain the entire set even if product is upgraded.
  - Use this to compare to the migrated version to detect changes you need to accommodate.

  We also recommend that you use the following strategy:

- Put only custom InputProcessors and PipelineComponents in the project source tree and leave all other stock dependencies in the WLCSPS code base.
- However, if custom IPs or PCs replaced stock components you used which have changed, then you must retrofit your custom components to fit the new product versions.

Migration Steps

Significant changes have taken place in this release. Be sure to progress through the migration as indicated in this document, for a successful migration. Keep in mind any relevant information from "Important Migration Tips and Strategies" on page 3-2 as you complete each step.

1. Be sure you've already reviewed the information in Chapter 1, "Planning Your Migration," beginning on page 1-1.
2. You must review "What's New in 3.5" on page 3-4 section before you begin, in order to be aware of the changes that have been made. This section contains brief descriptions of the changes in version 3.5, which you will need to address in order to migrate your system.
3. Then continue to the "Migrating Code from 3.2 to 3.5" on page 3-6 for instructions on migrating. This section contains detailed information about new items you might want to use, as well as changes to make to existing code.
4. Follow the instructions in "Migrating Data from 3.2 to 3.5" on page 3-13 to migrate your *database schemas*, as well as your *rulesets*.
5. Read "Migrating JSP Tags" on page 3-10 to find out the new tags in this release, as well as changes and how to migrate existing tags to the new release.
6. Read "Verifying the Migration to 3.5" on page 3-22 to ensure that configuration information is set up appropriately for this release.
7. Follow the steps in "Verifying the Migration to 3.5" on page 3-22 to make sure that the migration has happened correctly, and for tips on what to do if you encounter errors.
8. See "What's Next: Getting Started With 3.5" on page 3-23 for directions on information you might want to review to assist in development, performance, or understanding new features.

What's New in 3.5

This section covers the notable changes to this release. Be sure to review all of them, and make any changes necessary, based on how the changes affect your implementation. The information is organized as follows:

- Support for WebLogic Server 6.0
- Introducing the E-Business Control Center
- Changes to the Rules Editor in Release 3.5

Support for WebLogic Server 6.0

The Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server products now support WebLogic Server 6.0. The support for this new version of WebLogic Server includes a WebLogic Server domain, which contains the WebLogic Server Administration Console and the sample Web applications. A utility to migrate properties from the weblogic.commerce file to the new WebLogic Server Administration Console is provided, and the sample Web applications are reorganized and deployed in a single Enterprise Application. There is a new directory structure for Commerce Server 3.5, based on the WebLogic Server 6.0 directory structure. In addition, all Web-based pages are now required to be deployed as a Web application.

Introducing the E-Business Control Center

This release introduces the E-Business Control Center, a GUI tool designed to simplify the way business professionals manage their online customer relationships. This 100% Java client is a powerful desktop application that is easy to use and makes sense to business users by interacting with them in their own terms. The E-Business Control Center replaces the Rules Manager as the mechanism for creating and editing rules. Specialized versions of the E-Business Control Center are packaged with the Commerce Server and the WebLogic Personalization Server products.

For more information, see the *Guide to Using the E-Business Control Center* at http://edocs.bea.com/wlcs/docs35/campusr/index.htm.

Changes to the Rules Editor in Release 3.5

The new Personalization Rules Manager allows business users to fine-tune user-system interactions using plain-English commands within easy-to-use rule editing templates. The Personalization Rules Manager drives BEA's embedded rules engine and eliminates the need to master complex Boolean logic to create and edit rules.

The XML format (expressed in XMLSchema) used to describe rule sets for the 3.2 release of WebLogic Commerce Server has been replaced by a more manageable XML format designed to grow with future needs. A new tool, the E-Business Control Center, has also been introduced to allow you to more easily enter rules. This change enables you to specify a greater range of rules.

Migrating the rules for use with the new format is covered in "Migrating Your Rules" on page 3-18, within the "Migrating Data from 3.2 to 3.5" section.

Custom Tags

The following changes were made to JSP tag libraries in Release 3.5. See "Migrating JSP Tags" on page 3-10 for more information, and directions for migrating your implementation to accommodate these changes.

- Removed and Changed Tags

  `<es:preparedStatement>`

  `<pz:div>`

  `<pz:contentSelector>`

- New Tags

  `<ad:adTarget>`

  `<ph:placeholder>`

```
<tr:clickContentEvent> Content Tag
<tr:displayContentEvent> Content Tag
<trp:clickProductEvent> Product Tag
<trp:displayProductEvent> Product Tag
<trc:clickCampaignEvent> Campaign Tag
<webflow:setValidated Value>
<eb:smnav>
```

Migrating Code from 3.2 to 3.5

Use the information in this section to migrate the code, including JSP tags. The information in this section is organized as follows:

- Migrating Your System for Use With WebLogic Server 6.0
- Replacing the Rules Manager With the E-Business Control Center
- Migrating JSP Tags

Migrating Your System for Use With WebLogic Server 6.0

The Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server products now support WebLogic Server 6.0. The support for this new version of WebLogic Server includes a WebLogic Server domain, which contains the WebLogic Server Administration Console and the sample Web applications. A utility to migrate properties from the `weblogic.commerce` file to the new WebLogic Server Administration Console is provided, and the sample Web applications are reorganized and deployed in a single Enterprise Application.

Update Directory Structure Based on Changes to the WebLogic Commerce Server Directory Structure

New features and improvements to WebLogic Server 6.0 have caused changes in the Commerce Server 3.5 directory structure. This section shows the old and new directory structures and points out key changes. Be sure to examine the installation directory once you have installed WebLogic Commerce Server, to become familiar with the changes.

Figure 3-1 WebLogic Commerce Server 3.2 Sample Directory Structure

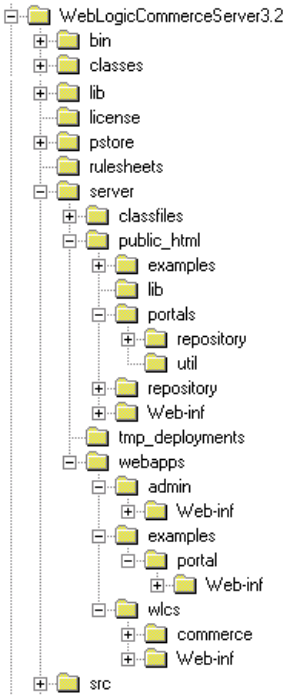
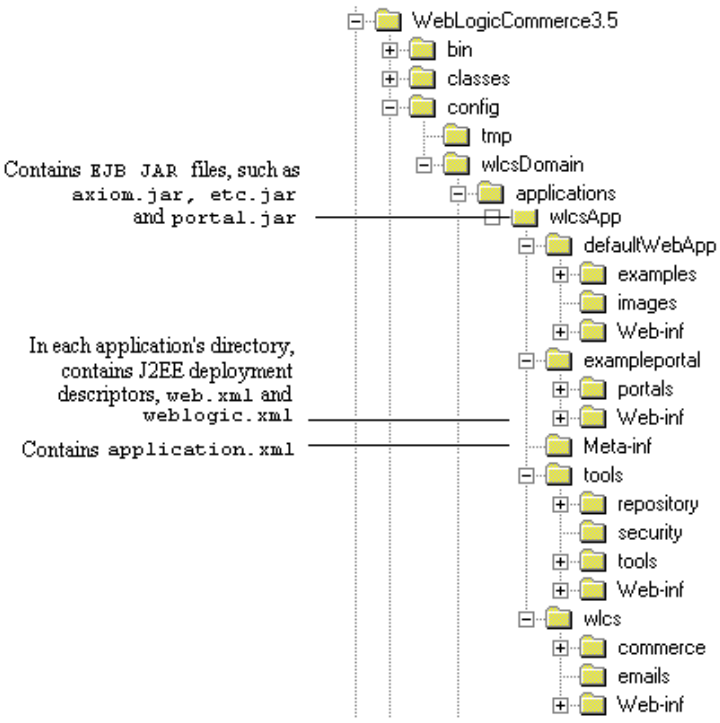


Figure 3-2 WebLogic Commerce Server 3.5 Sample Directory Structure



Key changes in the 3.5 WebLogic Commerce Server include the following:

- The `wlcsDomain` directory is the domain directory; the `wlcsDomain\applications` directory contains all other files and directories listed here.
- The `wlcsApp` directory contains the WebLogic Commerce Server application, including EJB JAR files.
- The `wlcsApp\META-INF` directory contains the `application.xml` file, the J2EE application deployment descriptor.
- The `WEB-INF` directory in each Web application directory contains deployment descriptors.

- The `wlcsApp\defaultWebApp` directory replaces `public_html` as the default Web application directory. You can specify one application as the default.
- Additional Web applications are stored at the same level as `wlcsApp\defaultWebApp`. The `tools` and `wlcs` folders are shown as examples.

For information about migrating the WebLogic Server, refer to the documentation for WebLogic Server 6.0.

Put All Pages in a Web Application

You must now deploy all Web-based pages as a Web application.

The direction of the product is to deploy one portal per Web application and use the application deployment features enabled through the WLS console, instead of the hot deployment portal model of the previous release. When migrating existing portals built on previous releases of the product, convert your non-Web applications to Web applications. This will result in one portal per Web application. You can use the out-of-the-box `exampleportal` as a model.

WebLogic Server supplies a default Web application for simple pages that do not require deployment descriptor properties.

For more information, see the section “Deploying New Portals as Web Applications” in the Guide to Creating Portals and Portlets.

Replacing the Rules Manager With the E-Business Control Center

This release introduces the E-Business Control Center, a GUI tool designed to simplify the way business professionals manage their online customer relationships. This 100% Java client is a powerful desktop application that is easy to use and makes sense to business users by interacting with them in their own terms.

The E-Business Control Center replaces the Rules Manager as the mechanism you should use for creating and editing rules. Using the tool, business users can now create their own customer segments (classification rules) and content selectors (content selector rules).

Specialized versions of the E-Business Control Center are packaged with the Commerce Server and the WebLogic Personalization Server products. Or you might choose to license the new Campaign Manager for WebLogic product, which includes the most comprehensive version of the E-Business Control Center currently available.

For more information, see the *Guide to Using the E-Business Control Center* at http://edocs.bea.com/wlcs/docs35/campusr/index.htm.

Migrating JSP Tags

This section covers the changes to JSP tags in this release, and what you need to accommodate those changes. Update your use of these tags as indicated by the changes described in this section, and review code that references these tags to be sure you have addressed all dependencies.

- Changed or Removed Tags
  - Removed Tags
  - Changes to Personalization Tags
- New Tags
  - New Ads and Placeholder Tag
  - New Event Tracking Tags
  - New Webflow Tag
  - New E-Business Tag

Changed or Removed Tags

Removed Tags

One tag has been removed in this release, `<es:preparedStatement>` tag.

Note: All tags that were deprecated in previous releases have now been removed.

Changes to Personalization Tags

The `ruleSet` attribute has been removed from the `<pz:div>` tag and the `<pz:contentSelector>` tag. This attribute was used to define the URI for the rulesets. In code that already used the `ruleSet` attribute, the attribute is no longer required and will be ignored.

It is no longer necessary for programmers to define rule sets (or rulesheets) because rule set names are no longer controlled through the tags. Rules are created using the new GUI tool, E-Business Control Center. The tool saves rules into predefined rule sets in the advislet registry. User classifier rules are saved into the `GlobalClassification.xml` file. Content selectors are saved into the `GlobalContentSelectors.xml` file.

New Tags

New Ads and Placeholder Tag

Two new tags have been added to create placeholders and query for ad content. These tags are documented in the "Personalization Server JSP Tag Library Reference" chapter in the *Guide to Building Personalized Applications*.

- <ad:adTarget> – The `<ad:adTarget>` tag uses the Ad Service to send an ad query to the content management system. Unlike the tag, the query in the `<ad:adTarget>` tag does not compete with other queries in an ad placeholder.
- <ph:placeholder> – The `<ph:placeholder>` tag implements a placeholder, which describes the behavior for a location on a JSP page.

New Event Tracking Tags

Five new JSP tags have been added to track events. These `<tr*:>` tags are documented in the chapter "Events and Behavior Tracking JSP Tag Library Reference" in the *Guide to Events and Behavior Tracking*.

- <tr:clickContentEvent> Content Tag – The `<tr:clickContentEvent>` tag is used to generate a behavior event when a user has clicked (through) on an ad impression.

- <tr:displayContentEvent> Content Tag – The `<tr:displayContentEvent>` tag is used to generate a behavior event when a user has received (viewed) an ad impression, (typically a `.gif` image).
- <trp:clickProductEvent> Product Tag – The `<trp:clickProductEvent>` tag is used to generate a behavior event when a user has clicked (through) on a product impression. This tag will return a URL query string containing event parameters. It is then used when forming the complete URL that hyperlinks the content.
- <trp:displayProductEvent> Product Tag – The `<trp:displayProductEvent>` tag is used to generate a behavior event when a user has received (viewed) a product impression (typically a `.gif` image).
- <trc:clickCampaignEvent> Campaign Tag – The `<trc:clickCampaignEvent>` tag is used to explicitly generate a clickthrough event relevant to a campaign.

New Webflow Tag

- <webflow:setValidated Value> – The tag `<webflow:setValidatedValue>` is used in a JSP to configure the display of fields in a form that a customer must correct.

The Pipeline and Webflow tags are documented in the chapter "Webflow and Pipeline JSP Tag Library Reference" in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*.

Several previously undocumented attributes have been added to the documentation for the `<webflow:getValidatedValue>` tag.

New E-Business Tag

- <eb:smnav> – The `<eb:>` preface stands for E-Business. The Scrollable Model can be use throughout the E-Business package to iterate through a list of objects. It can be used in conjunction with transaction, shopping cart, order history, or shipping services.

See the chapter "Product Catalog JSP Tag Library Reference" in the *Guide to Building a Product Catalog*.

Migrating Data from 3.2 to 3.5

This section covers upgrading database schemas and rules from 3.2 to 3.5.

Release 3.5 of WebLogic Commerce Server and WebLogic Personalization Server provides enhancements and changes that require you to update the schemas and migrate the data.

Note: If you are using the Oracle 8.0.5 or 8.1.5 database, you must upgrade to Oracle 8.1.6 or greater before migrating to Release 3.5 of WebLogic Commerce Server and WebLogic Personalization Server. Release 3.5 uses CLOBs and BLOBs instead of LONG RAW characters. Oracle 8.0.5 and 8.1.5 do not support CLOBs and BLOBs.

Complete the tasks in the following sections:

Schema Migration Steps

- Make a Backup
- Validate Data
- Upgrade Current Tables to New Schema
- Drop Any Backup Tables (Optional)
- Add New Tables to Bring the Schema Current
- Verify the Upgrade

Rules Migration Steps

- Migrating Your Rules

Note: Information about migrating specific databases is also provided in readme files. Navigate to the readme file for your specific database using this directory path:

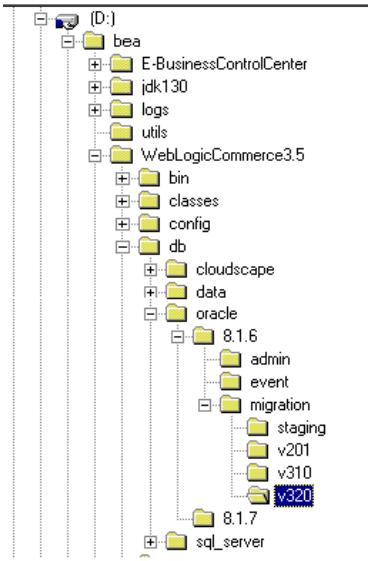
```
../db/<vendor>/<version>/migration/v320/readme.text
```

where <vendor> is the name of the database (for example, Oracle) and

<version> is the release number for that database.

Figure 3-3 shows the directory structure on a Windows system.

Figure 3-3 Navigate to the Database readme Files



Make a Backup

We strongly recommend that you make a complete backup of the WebLogic Personalization Server/WebLogic Commerce Server database before beginning the process to migrate the database to a more current schema.

Validate Data

To validate the data before beginning the upgrade, run the following scripts. These scripts are used to check which WebLogic Commerce Server tables have column values exceeding 254 characters, or whatever the new length for the specific column in question is.

1. From a command prompt, type the following:

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/check_common_lengths.sql
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/check_wlps_lengths.sql
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/check_wlcs_lengths.sql
```

For example, if you are using Oracle 8.1.6 and installed Commerce Server in `~/WebLogicCommerceServer3.5`, enter the following command in SQL*Plus:

```
@ $WL_COMMERCE_HOME/db/oracle/8.1.6/migration/v320/check_common_lengths.sql
```

2. To see the results of the script, open the following log file in a text editor:

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/check_common_lengths.log
```

There will also be `check_wlps_lengths.log` and `check_wlcs_lengths.log`.

The log file lists each table for which the maximum number of characters has changed. As Listing 3-1 illustrates, the log file states `no rows selected` for tables that meet the new maximum-length requirements. For tables that exceed requirements, the log file lists each row and describes the error condition.

Listing 3-1 Output of check-wlcs-lengths.sql

```
*****  WLCS_CATEGORY  *****
no rows selected

*****  WLCS_PRODUCT  *****
no rows selected
```

3. For any table containing data that exceeds a row's maximum length requirement:
   a. For information on length requirements for the table, see the schema chapters in the Commerce Server and WebLogic Personalization Server documentation.
   b. Modify the data in the row to meet the new requirements.
4. Repeat the previous steps until the log file for each script reports `no rows selected` for all tables.

Upgrade Current Tables to New Schema

Once you have successfully verified the current tables, you can run the upgrade scripts. These scripts make backup copies of current tables, drop the existing table, recreate each table with current column definitions, and populate the new table with data from the backup table.

From a command prompt, type the following:

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/upgrade_common_to_350.sql
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/upgrade_wlps_to_350.sql
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/upgrade_wlcs_to_350.sql
```

Drop Any Backup Tables

Note: This step is optional and should not be executed until you are certain that all data has been retained. Only when you are comfortable with the state of your data should you consider running this script.

The script in this step is used to drop the temporary tables that held your original data. During the upgrade process, the data contained in the original table (named with the WLCS_ prefix) was copied to a backup table using the BEA_ prefix and renamed with BEA_ prefix.

1. Restart the WebLogic Commerce Server before executing this script, to test data accuracy and to ensure the application(s) are working as intended.
2. From a command prompt, type the following:

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/drop_bea_tables.sql
```

Add New Tables to Bring the Schema Current

Run the following scripts to create the necessary new tables.

- `insert_event_properties.sql` is used to populate the property management tables with records of various Behavior Tracking events.

- `upgrade_wlcs_add_tables_350.sql` is used to create new tables used by the WebLogic Commerce Server. It is involved in making changes to the existing database objects: adding columns, dropping columns, resizing columns, and so on.
- `create_campaign.sql` and `create_mail_ad.sql` are used in the new WebLogic Campaign Manager component. Each script is involved in creating new database objects such as tables and indexes.

From a command prompt, type the following:

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/
upgrade_wlcs_add_tables_350.sql
```

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/migration/v320/
insert_event_properties.sql
```

Note: The path from which to run the next two scripts is different from where you ran previous scripts.

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/create_campaign.sql
```

```
@ $WL_COMMERCE_HOME/db/db-vendor/db-version/create_mail_ad.sql
```

Verify the Upgrade

After you upgrade the schema for each server that you are using, verify the upgrade by starting the server and Administration Tool and testing the application. For example, if you use both Commerce Server and WebLogic Personalization Server, open the Administration Tool to verify that the users and groups you upgraded are available under User Administration, and all items and categories that you upgraded are available under Catalog Administration. Then access the server through a Web browser to verify that data transferred successfully.

Starting the Server

To start Commerce Server and/or WebLogic Personalization Server on UNIX, enter the following command from a WebLogic Commerce Server and WebLogic Personalization Server host:

```
$WL_COMMERCE_HOME/StartCommerce.sh
```

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

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

Migrating Your Rules

The new Personalization Rules Manager allows business users to fine-tune user-system interactions using plain-English commands within easy-to-use rule editing templates. The Personalization Rules Manager drives BEA's embedded rules engine and eliminates the need to master complex Boolean logic to create and edit rules.

Note: This section assumes you are migrating from Release 3.2. If you are migrating from Release 3.1 or earlier, begin with the section "Changes to the Rules Editor in Release 3.1" in Chapter 5, "Migrating WebLogic Personalization Server From Version 2.0.1 to Version 3.1."

The XML format (expressed in XMLSchema) used to describe rule sets for the 3.2 release of WebLogic Commerce Server has been replaced by a more manageable XML format designed to grow with future needs. A new tool, the E-Business Control Center, has also been introduced to allow you to more easily enter rules. This change enables you to specify a greater range of rules.

To use your existing Web Logic Commerce Server 3.2 rules with the new format, follow the steps in this section.

Note: This procedure provides you with files listing all your rules, which you can print and use as a reference to reenter them. As an alternative to steps 1-6, you can view your rules from within the Rules Management JSP pages in WebLogic Commerce Server 3.2, and use that display as the template for the conversion. Then go to step 7 and follow the directions there.

1. Ensure that the correct systems are installed.
   - WebLogic Server 5.1 and the required service pack. See http://edocs.bea.com/wlcs/docs32/relnotes/relnotes.htm#platforms.

- WebLogic Commerce Server 3.2.
- WebLogic Server 6.0 and required service pack and rolling patch. See http://edocs.bea.com/wlcs/docs35/install/platforms.htm.
- Campaign Manager for WebLogic, WebLogic Commerce Server, or WebLogic Personalization Server 3.5.
- E-Business Control Center, which contains the correct rules parser.

2. Install WebLogic Commerce Server 3.5. WLCS must be both *installed* and *running*. Installing WebLogic Commerce Server 3.5 provides the files used during the migration; verify that the files are there before you continue. The paths to the files on your computer might be slightly different, but the filenames must be the same.

   `<wlcs3.5 install dir>/bin/win/dumprules32.bat` (Windows only)

   `<wlcs3.5 install dir>/bin/unix/dumprules32.sh` (UNIX only)

   `<wlcs3.5 install dir>/classes/rules-tools-common.properties`

   `<wlcs3.5 install dir>/classes/rules-tools-query-32.properties`

   `<wlcs3.5 install dir>/classes/rules-tools-dump.properties`

   `<wlcs3.5 install dir>/classes/com/bea/commerce/platform/rules/tools/RuleSetProcessorHarness.class`

   `<wlcs3.5 install dir>/classes/<supporting program classes>`

3. Edit the files shown in Table 3-1 to let the migration script know where your WLS installation, existing rules, and other relevant files are located.

Table 3-1 Files and Corresponding Variables for Rules Migration

File	Variable	Variable Value
`dumprules32.bat` or `dumprules32.sh`	`WLS_51_HOME`	The root directory of an existing WLS 5.1 installation, such as `/opt/bea/wlserver5.1`
	`WLCS_35_TOOLS_LIB_EXT`	The `<root>/lib/ext` directory of the Campaign Manager tool installation, such as `/opt/bea/WebLogicCommerceServer3.5/tools/lib/ext`

Table 3-1 Files and Corresponding Variables for Rules Migration (Continued)

File	Variable	Variable Value
rules-tools-query-32.properties	t3-host	The name of a running WLCS 3.2 installation host, in the following format: Syntax: http://<t3-host>:<t3-port> Example: http://localhost:7501
	t3-port	The name of a running WLCS 3.2 installation port, in the following format: Syntax: http://<t3-host>:<t3-port> Example: http://localhost:7501

4. Use the dumprules32 script to create files containing lists of your rules. This script creates two files for each of your rule sets in your existing database.

   You will use one of them as a reference to reenter the rules in the new format.

   a. Determine a location (directory) on the local filesystem to write out 3.2 rule sets, such as C:\rules32.

   b. Manually create that directory. For example, in Windows on the command line you would type C:\> mkdir rules32.

   c. Run the dumprule32 script from within the home directory of the script, and specify the directory you created in the previous step. For example, in Windows you would run the following command from the command line:

```
C:\opt\bea\WebLogicCommerceServer3.5\bin\win32> dumprules32.bat C:\rules32
```

   Two files are written for each rule set:

   <*rule set name*>.ruleset—contains the rule sets in XML.

   <*rule set name*>-txt.ruleset—contains the rule sets in a more readable text format.

   You will use the -txt.ruleset file for each rule set to reenter your rules.

5. Go to the directory you specified when you ran the script. Print each `-txt.ruleset` file that was created, or open them in a text application, so that they are ready for you to use them.

6. Review the printed file so that you can easily determine what each rule states from the printout. The following is a partial example of a `-txt.ruleset` file.

```
-- Begin File AcmeRules-txt.ruleset --
(RuleSet)
Name: SampleRuleSet
Description: Demonstrating And/or Logic in Rules Conversion
(Rule)
       Name: SampleRule1
       Type: classifier
          (When)
             REQUEST.DefaultRequestPropertySet.Authorization Scheme eq 1
                (Or)
                   REQUEST.DefaultRequestPropertySet.Character Encoding eq 2
                   REQUEST.DefaultRequestPropertySet.Character Encoding gt 5
          (Then)
             (New)
                ClassName: Classification
                   (Arguments)
                   (Constant)
                   Type: string
                   Value: RuleDemonstrated
```

You need to pay particular attention to the lines under the (`When`) heading in order to recreate your rules correctly, if there are lines under (`When`) as well as (`Or`). Both the line after the (`When`) and the *first* line after the (`Or`) must be true for the logic under (`Then`) to be run.

Note: The rule set printout makes it seems as though either the (`When`) item or the first (`Or`) item could be true to fulfill the condition; however, that is not the case. As stated previously, both must be true.

For example, in the sample rule set given, the string "`RuleDemonstrated`" will be displayed in either of the following situations:

If `DefaultRequestPropertySet.Authorization Scheme` equals 1 *and* `REQUEST.DefaultRequestPropertySet.Character Encoding` equals 2

or

If `DefaultRequestPropertySet.Authorization Scheme` equals 1 *and* `REQUEST.DefaultRequestPropertySet.Character Encoding` is greater than 5

The following would *not* fulfill the requirements for the condition (`Then`):

If `DefaultRequestPropertySet.Authorization Scheme` equals 1 **or** `REQUEST.DefaultRequestPropertySet.Character Encoding` is greater than 5

7. Enter the new rules in the E-Business Control Center. For information about using the E-Business Control Center, see *Using the BEA E-Business Control Center*.

Note: Rules are created in the E-Business Control Center. This GUI tool is designed to allow Business Analysts to develop their own content selector rules and classifier rules. Because the Business Analysts are not exposed to the concept of rules, you will see content selector rules called simply "content selectors" and classifier rules referred to as "customer segments."

For more information about rules, see the following chapters in the *Guide to Building Personalized Applications*:
Creating Personalized Applications with the Advisor
Foundation Classes and Utilities
Introducing the Rules Manager
Working with Content Selectors

Migrating Your WLCS License

There is a new licensing upgrade step in this release, to convert your older Commerce Server license to the new XML format, refer to the installation guide:

http://e-docs.bea.com/wlcs/docs35//install/postinst.htm

Verifying the Migration to 3.5

In a test environment, run a typical selection of common user and administrator tasks to be sure your system was migrated correctly. Suggested tasks include the following:

- Go to the example portal and be sure you can log in.

- Go to the example Commerce Server application and make sure you can both view items and add them to a shopping cart.
- Log into the administration tools and view properties for several users.

Refer to the appropriate section of the referenced documentation and this guide if you encounter errors.

What's Next: Getting Started With 3.5

Use the online documentation site to locate the documentation you need for additional learning.

http://edocs.bea.com/wlcs/docs35/index.html

For an overview of how the products work together, see the *Product Family Overview* at http://edocs.bea.com/wlcs/docs35/intro/index.htm.

You might also want to review the following guides, new in this release:

- *Guide to Developing Campaign Infrastructure*, available at http://edocs.bea.com/wlcs/docs35/campdev/index.htm
- *Security Guide*, available at http://edocs.bea.com/wlcs/docs35/secguide/index.htm
- *Performance and Tuning Guide*, available at http://edocs.bea.com/wlcs/docs35/tuning/index.htm
- *Deployment Guide*, available at http://edocs.bea.com/wlcs/docs35/deploygd/index.htm

CHAPTER

4 Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2

This chapter describes the changes between WebLogic Commerce Server and WebLogic Personalization Server Release 3.2 and the previous release, 3.1.1.

Note: You typically must migrate from one release to the next consecutive one. However, there is one exception. To migrate WebLogic Personalization Server from 2.0.1 to 3.2, see "2.0.1 to 3.2: WebLogic Personalization Server" on page 4-9, within "Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2."

For migrating from 2.0.1 to 3.1.1, see "Migrating WebLogic Personalization Server From Version 2.0.1 to Version 3.1" on page 5-1.

The WebLogic Personalization Server is bundled with the Commerce services. This document uses "Commerce services" to refer to both servers.

This section includes the following topics:

- Migration Checklist for 3.1.1 to 3.2
- What's New in 3.2
- Migrating Code From 3.1.1 to 3.2

- Migrating Data From 3.1.1 to 3.2
- Verifying the Migration to 3.2
- What's Next: Getting Started With 3.2

Migration Checklist for 3.1.1 to 3.2

Follow the steps in this section to migrate your system to the current release.

1. Be sure you've already reviewed the information in Chapter 1, "Planning Your Migration," beginning on page 1-1.
2. The bulk of the code migration consists of your reviewing the new and changed features, locating where in your code you use or depend on any of them, and making the appropriate changes. Review "What's New in 3.2" on page 4-3 so you know the main types of issues involved; the subsequent migration sections provide more detail and instructions.
3. Follow the steps in "Migrating Code From 3.1.1 to 3.2" on page 4-6 to make the necessary changes to your code.
4. Follow the steps in "Migrating Data From 3.1.1 to 3.2" on page 4-8 to migrate database schemas.

Note: To upgrade from 2.0.1 to 3.2, see "Migrating Data" on page 5-85 of *Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2*.

5. Read "Verifying the Migration to 3.2" on page 4-17 to learn details about new custom tags, and to make the necessary changes to tags that have been changed.
6. Follow the steps in "Verifying the Migration to 3.2" on page 4-17 to ensure that your migration was completed successfully.
7. Read "What's Next: Getting Started With 3.2" on page 4-18 for suggestions on next steps and additional information.

What's New in 3.2

Note: Be sure to also read the "What's New" page on the e-docs.bea.com Web site for a preview of the new features in this release: http://e-docs.bea.com/wlcs/docs32/interm/whatsnew.htm

This section covers the notable changes to this release. Be sure to review all of them, and make any changes necessary, based on how the changes affect your implementation. The information is organized as follows:

- New Java 2 SDK for Enhanced Performance
- New Third-Party Integrations
- New Webflow and Pipeline Editor
- Custom Tags

New Java 2 SDK for Enhanced Performance

The BEA Commerce services and WebLogic Personalization Server now require the Java 2 SDK with the HotSpot Server Virtual Machine. Using the Java 2 SDK with HotSpot may enhance performance for your production environment.

New Third-Party Integrations

BEA WebLogic Commerce is now integrated with the following third-party products:

- International Tax Support from TAXWARE
- E-Marketing Analysis Using Broadbase
- Content Management with Interwoven Content Express

International Tax Support from TAXWARE

The WORLDTAX System from TAXWARE International, Inc. is the most comprehensive calculation system for international taxes available in the industry. The WORLDTAX System calculates and reports Value Added Tax (VAT), Goods and Services Tax (GST), sales tax, and consumption tax in many countries. BEA has tested the countries of France, Germany, Italy, South Korea, Spain, and the United Kingdom for accuracy with the WebLogic Commerce Server.

E-Marketing Analysis Using Broadbase

This release of the BEA WebLogic Commerce and Personalization Server provides integration with Broadbase for customer analysis. The user profile, customer, and order information can be extracted from the WebLogic Commerce and Personalization Server database and used to enable E-Marketing analysis within Broadbase. Please contact Broadbase Software, Inc. for more information.

Content Management with Interwoven Content Express

In addition to the existing limited-user version of Documentum 4i, this release includes Interwoven Content Express, an entry-level content management tool. These content management products allow developers to quickly create personalized content applications, using software from either of the leading content management vendors.

New Webflow and Pipeline Editor

The BEA WebLogic Commerce Server now includes the Webflow and Pipeline Editor, a JSP-based administration tool specifically designed to help you modify the default `webflow.properties` and `pipeline.properties` configuration files. It also provides you with validation tools that enable you to check the syntax of your Webflow and verify whether the necessary components exist within the Webflow. By modifying and validating your Webflow with the Webflow and Pipeline Editor, you can eliminate errors that may otherwise be difficult to track.

Custom Tags

Note: **Backward Compatibility Will Stop After Version 3.2.** The tag libraries were updated in WebLogic Personalization Server (WLPS) version 3.1 to comply with the JSP 1.1 Specification. If you are upgrading from WebLogic Personalization Server 2.0.1, you can continue to use your existing code with WebLogic Personalization Server 3.2. However, *future releases will no longer be backward compatible*, so you will need to migrate to the new tags if you intend to continue to use your legacy code with the latest WebLogic Personalization Server releases.

New Custom Tags

WebLogic Personalization Server 3.2 introduces eight new tags:

- <cm:getProperty> – Retrieves the value of the specified content metadata property into a variable specified by `resultId`. This tag is similar to the `<cm:printProperty>` tag, with the addition of two new parameters, `resultId` and `resultType`.
- <fm:getApplicationURI> – Gets the Flow Manager.
- <fm:getCachedAttribute> – Gets an attribute out of the session/global cache.
- <fm:setCachedAttribute> – Sets an attribute in the session/global cache.
- <fm:removeCachedAttribute> – Removes an attribute from the session/global cache.
- <fm:getSessionAttribute> – Gets an attribute out of the HttpSession.
- <fm:setSessionAttribute> – Sets an attribute in the HttpSession.
- <fm:removeSessionAttribute> – Removes an attribute from the HttpSession.

Changed Custom Tags

- Changes to Content Management Tags in Release 3.2 – > – A new attribute, `baseHref`, has been added to the `<cm:printDoc>` tag. This attribute provides the URL of the document's BASE HREF.

- Changes to Utility Tags in Release 3.2 – The Personalization Utility tag `<es:preparedStatement>` has a new attribute, `transactionIsolationLevel`.

Note: This tag is removed from the tag library in WLPS Release 3.5.

Migrating Code From 3.1.1 to 3.2

Use the information in this section to migrate the code, including JSP tags. The information in this section is organized as follows:

- Use New Java 2 SDK for Enhanced Performance
- Use New Webflow and Pipeline Editor
- New JSP Tags and Migrating Existing JSP Tags

Use New Java 2 SDK for Enhanced Performance

The BEA Commerce services and WebLogic Personalization Server now require that you use Java 2 SDK with the HotSpot Server Virtual Machine. Using the Java 2 SDK with HotSpot may enhance performance for your production environment.

Use New Webflow and Pipeline Editor

The BEA WebLogic Commerce Server now includes the Webflow and Pipeline Editor, a JSP-based administration tool specifically designed to help you modify the default `webflow.properties` and `pipeline.properties` configuration files. It also provides you with validation tools that enable you to check the syntax of your Webflow and verify whether the necessary components exist within the Webflow. Modify and validate your Webflow with the Webflow and Pipeline Editor in order to eliminate errors that may otherwise be difficult to track.

New JSP Tags and Migrating Existing JSP Tags

This section covers the new and changed JSP tags in this release.

- Changed Custom Tags
- New Custom Tags

Changed Custom Tags

Update your use of these tags as indicated by the changes described in this section, and review code that references these tags to be sure you have addressed all dependencies.

Note: **Backward Compatibility Will Stop After Version 3.2.** The tag libraries were updated in WebLogic Personalization Server (WLPS) version 3.1 to comply with the JSP 1.1 Specification. If you are upgrading from WebLogic Personalization Server 2.0.1, you can continue to use your existing code with WebLogic Personalization Server 3.2. However, *future releases will no longer be backward compatible*, so you will need to migrate to the new tags if you intend to continue to use your legacy code with the latest WebLogic Personalization Server releases.

The WebLogic Personalization Server documentation has been revised to reflect the changes to the tag libraries. Until you migrate to the new tags, you can continue to use the WebLogic Personalization Server 2.0 *JSP Tag Reference* found at http://e-docs.bea.com/wlcs/docs20/p13ndev/jsptags.htm.

- Changes to Content Management Tags in Release 3.2 – > – A new attribute, `baseHref`, has been added to the `<cm:printDoc>` tag. This attribute provides the URL of the document's BASE HREF.
- Changes to Utility Tags in Release 3.2 – The Personalization Utility tag `<es:preparedStatement>` has a new attribute, `transactionIsolationLevel`.

Note: This tag is removed from the tag library in WLPS Release 3.5.

New Custom Tags

WebLogic Personalization Server 3.2 introduces eight new tags:

- <cm:getProperty> – Retrieves the value of the specified content metadata property into a variable specified by `resultId`. This tag is similar to the `<cm:printProperty>` tag, with the addition of two new parameters, `resultId` and `resultType`.
- <fm:getApplicationURI> – Gets the Flow Manager.
- <fm:getCachedAttribute> – Gets an attribute out of the session/global cache.
- <fm:setCachedAttribute> – Sets an attribute in the session/global cache.
- <fm:removeCachedAttribute> – Removes an attribute from the session/global cache.
- <fm:getSessionAttribute> – Gets an attribute out of the HttpSession.
- <fm:setSessionAttribute> – Sets an attribute in the HttpSession.
- <fm:removeSessionAttribute> – Removes an attribute from the HttpSession.

Migrating Data From 3.1.1 to 3.2

Release 3.2 of WebLogic Commerce Server and WebLogic Personalization Server introduces scripts that you can use to create Oracle tablespaces and user accounts for the WebLogic Commerce Server and WebLogic Personalization Server database schema. It also introduces scripts that you can use to export the Oracle database objects from a source environment and import them into a destination environment. With these export/import scripts, you can move data from a staging environment into your production environment without having to recreate all your content.

Release 3.2 introduces schema changes and restrictions for the length of data allowed in various columns. To upgrade databases from Release 3.1.1 to Release 3.2, complete the following tasks:

- 2.0.1 to 3.2: WebLogic Personalization Server
- 3.1.1 to 3.2: WebLogic Commerce Server and WebLogic Personalization Server

Note: If you are starting at a release earlier than Release 3.1.1 of WebLogic Commerce Server and WebLogic Personalization Server, begin at the appropriate section in that chapter. Database migration must be done sequentially. You cannot skip a release.

Information about migrating specific databases is also provided in readme files. Navigate to the readme file for your specific database using this directory path:

`../db/<vendor>/<version>/migration/v320/readme.text`

where `<vendor>` is the name of the database (for example, Oracle) and `<version>` is the release number for that database.

2.0.1 to 3.2: WebLogic Personalization Server

This section covers upgrading WebLogic Personalization Server database schemas from 2.0.1 to 3.2.

This section describes scripts that do the following:

- Update tables that were created for the WebLogic Personalization Server 2.0.1 schema to the WebLogic Personalization Server 3.2 schema.
- Copy existing WebLogic Personalization Server data to the new 3.2 tables.
- Create empty WebLogic Personalization Server tables that are new for 3.1.1 and 3.2.
- Create empty WebLogic Commerce Server tables that the release 3.2 schema defines. The scripts do not copy existing WebLogic Commerce Server data into the tables.

The following instructions also assume that you are working with an Oracle database and will be using the scripts in either of the following:

- `WL_COMMERCE_HOME/db/oracle` (for Oracle versions 8.1.5 and below)
- `WL_COMMERCE_HOME/db/oracle816` (for Oracle version 8.1.6)

To upgrade your Personalization Server database from 2.0.1, do the following:

1. In `WL_COMMERCE_HOME/db/<oracle directory>/migration/v201`, log into SQL*Plus and execute the following:

```
SQL> @upgrade-to-310.sql
```

2. In `WL_COMMERCE_HOME/db/<oracle directory>`, log into SQL*Plus and execute the following:

   ```
   SQL> @create-wlcs-oracle.sql
   ```

   At this point the database format should be WLCS 3.1.1 compliant.

3. Follow the steps in "3.1.1 to 3.2: WebLogic Commerce Server and WebLogic Personalization Server" on page 4-10. Then continue to step 4.

4. Start WebLogic Personalization Server and WebLogic Commerce Server if it is not already running.

5. In `WL_COMMERCE_HOME\bin\win32\` (Windows) or in `WL_COMMERCE_HOME/bin/unix/` (UNIX), run the following on a command line:

   ```
   loadrules
   ```

6. Finally, in `WL_COMMERCE_HOME\bin\win32\` (Windows) or in `WL_COMMERCE_HOME/bin/unix/` (UNIX), run the following on a command line:

   ```
   loaddocs -delete -cleanup
   ```

3.1.1 to 3.2: WebLogic Commerce Server and WebLogic Personalization Server

This section covers upgrading database schemas from 3.1.1 to 3.2

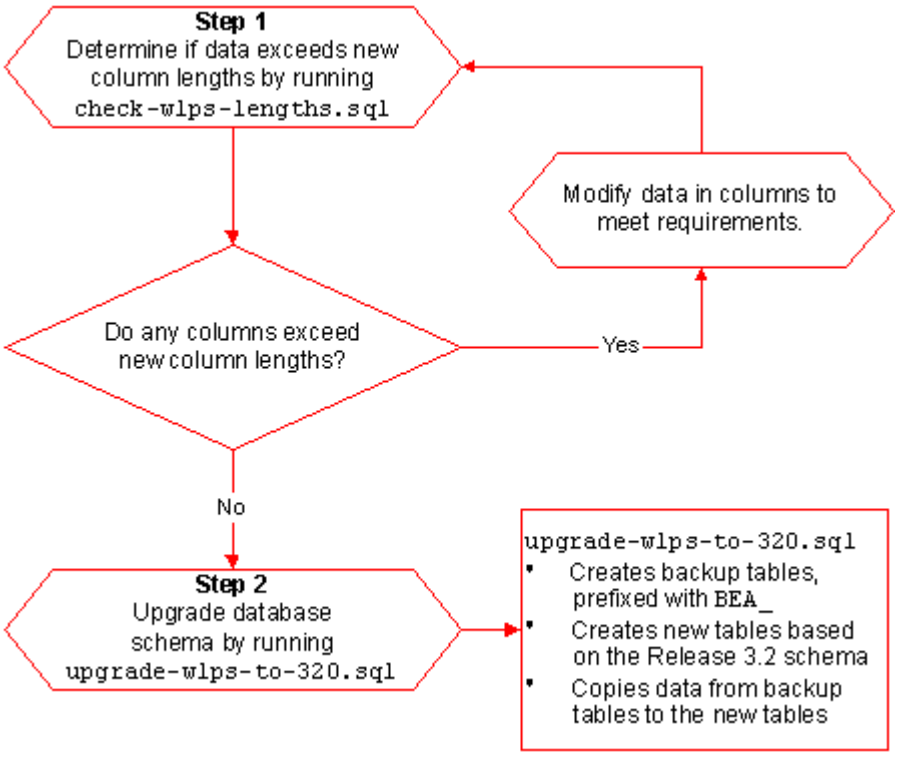
Release 3.2 of WebLogic Commerce Server and WebLogic Personalization Server introduces schema changes and restrictions for the length of data allowed in various columns. To upgrade databases from Release 3.1.1 to Release 3.2, complete the following tasks:

- Upgrade the WebLogic Personalization Server Schema
- Upgrade the Commerce services Schema (only if you use Commerce services)
- Verify the Upgrade
- Remove Temporary Tables

Upgrade the WebLogic Personalization Server Schema

If you are upgrading WebLogic Personalization Server from Release 3.1.1 to Release 3.2, complete the tasks described in this section. The following diagram illustrates the process for upgrading the WebLogic Personalization Server schema, and subsequent topics provide more information about the process.

Figure 4-1 Upgrading the Personalization Server Schema from 3.1.1 to 3.2



Step 1: Determine if Data Exceeds New Column Lengths and Modify When Necessary

Start the migration process by finding and correcting any columns in your existing databases that contain data exceeding the new column length in Release 3.2.

To start the migration, do the following:

1. Make a backup copy of your database.

2. Run the following SQL command:
   `@ $WL_COMMERCE_HOME/db/database-type/check-wlps-lengths.sql`

   For example, if you installed Commerce services in `~/WebLogicCommerceServer3.2`, enter the following command in SQL*Plus:
   `@ ~/WebLogicCommerceServer3.2/db/database-type/check-wlps-lengths.sql`

3. To see the results of the script, open the following log file in a text editor:
   `$WL_COMMERCE_HOME/db/database-type/check-wlps-lengths.log`

   The log file lists each table for which the maximum number of characters has changed. As Listing 4-1 illustrates, the log file states `no rows selected` for tables that meet the new maximum-length requirements. For tables that exceed requirements, the log file lists each row and describes the error condition.

Listing 4-1 Output of check-wlps-lengths.sql

```
*****  WLCS_DOCUMENT.ID  *****
no rows selected

*****  WLCS_DOCUMENT_METADATA.ID  *****
no rows selected
```

4. For any table containing data that exceeds a row's maximum length requirement:

   a. For information on length requirements for the table, see the schema chapters in the Commerce services and WebLogic Personalization Server documentation.

   b. Modify the data in the row to meet the new requirements.

5. Repeat steps 2-4 until the log file reports "no rows selected" for all tables.

Step 2: Upgrade the Database Schema

After correcting any rows that do not conform to new column length requirements, you must upgrade the Release 3.1.1 schema to the Release 3.2 schema by doing the following:

1. Make backup copies of your database and the following file:
   `$WL_COMMERCE_HOME/db/database-type/upgrade-wlps-to-320.sql`

2. Open `upgrade-wlps-to-320.sql` in a text editor.
3. Make sure that the following lines assign values that match your tablespace:

   ```
   define DATA_TABLESPACE=WLCS_DATA
   define INDEX_TABLESPACE=WLCS_INDEX
   ```

   By default, WebLogic Commerce Server and WebLogic Personalization Server place data in `WLCS_DATA` and indexes in `WLCS_INDEX`. If you are using other tablespaces, you must modify `upgrade-wlps-to-320.sql` to specify your tablespaces instead.
4. Save your modifications to `upgrade-wlps-to-320.sql`.
5. Run the following SQL command:
   `@ $WL_COMMERCE_HOME/db/`*`database-type`*`/upgrade-wlps-to-320.sql`

Note: Enter this command only once and only after you have modified all rows that contain data exceeding new length requirements.

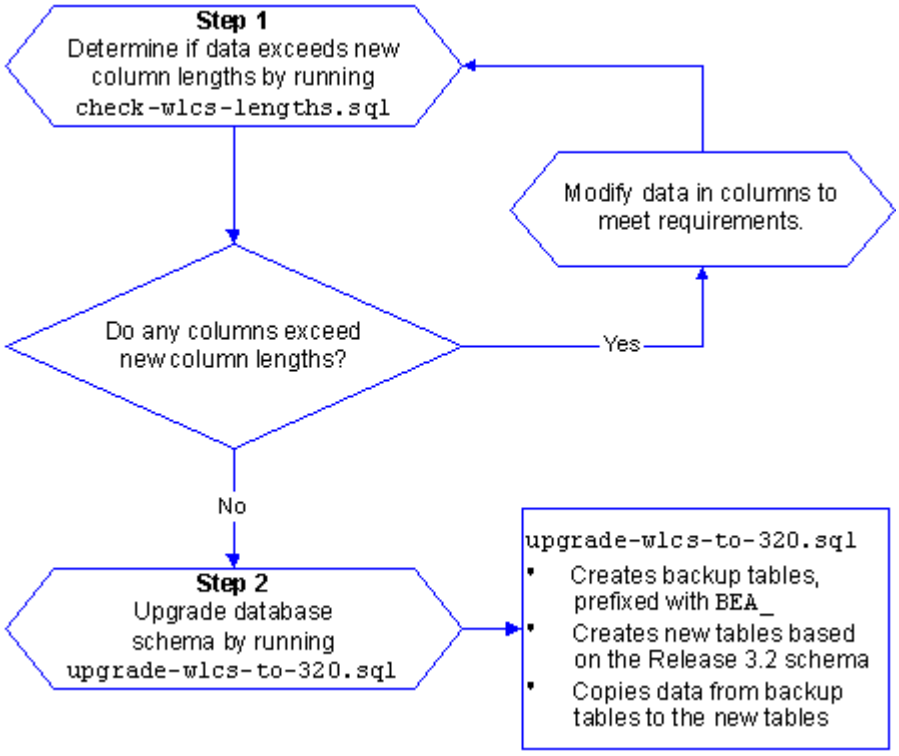
When the command successfully completes updating tables, it prints the following message:

```
************  SCRIPT HAS FINISHED EXECUTING   *************
******  PLEASE REVIEW UPDATE-TO-320.LOG  FILE  *****
```

Upgrade the Commerce services Schema

To upgrade Commerce services from Release 3.1.1 to Release 3.2, complete the tasks described in this section. The following diagram illustrates the process for upgrading the Commerce services schema, and subsequent topics provide more information about the process.

Figure 4-2 Upgrading the Commerce Server Schema from 3.1.1 to 3.2



Step 1: Determine if Data Exceeds New Column Lengths and Modify When Necessary

Start the migration process by finding and correcting any columns in your existing databases that contain data exceeding the new column length in Release 3.2.

To start the migration, do the following:

1. Make a backup copy of your database.
2. Run the following SQL command:
   `@ $WL_COMMERCE_HOME/db/`*`database-type`*`/check-wlcs-lengths.sql`
3. To see the results of the script, open the following log file in a text editor:
   `$WL_COMMERCE_HOME/db/`*`database-type`*`/check-wlcs-lengths.log`

The log file lists each table for which the maximum number of characters has changed. As Listing 4-2 illustrates, the log file states `no rows selected` for tables that meet the new maximum-length requirements. For tables that exceed requirements, the log file lists each row and describes the error condition.

Listing 4-2 Output of check-wlcs-lengths.sql

```
*****  WLCS_CATEGORY  *****
no rows selected

*****  WLCS_PRODUCT  *****
no rows selected
```

4. For any table containing data that exceeds a row's maximum length requirement:
   a. For information on length requirements for the table, see the schema chapters in the Commerce services and WebLogic Personalization Server documentation.
   b. Modify the data in the row to meet the new requirements.
5. Repeat steps 2-4 until the log file reports `no rows selected` for all tables.

Step 2: Upgrade the Database Schema

After correcting any rows that do not conform to new column length requirements, you must upgrade the Release 3.1.1 schema to the Release 3.2 schema by doing the following:

1. Make backup copies of your database and the following file:
   `$WL_COMMERCE_HOME/db/`*`database-type`*`/upgrade-wlcs-to-320.sql`
2. Open `upgrade-wlcs-to-320.sql` in a text editor.
3. Make sure that the following lines assign values that match your tablespace:

   ```
   define DATA_TABLESPACE=WLCS_DATA
   define INDEX_TABLESPACE=WLCS_INDEX
   ```

   By default, WebLogic Commerce Server and WebLogic Personalization Server place data in `WLCS_DATA` and indexes in `WLCS_INDEX`. If you are using other

tablespaces, you must modify `upgrade-wlps-to-320.sql` to specify your tablespaces instead.

4. Save your modifications to `upgrade-wlcs-to-320.sql`.
5. Run the following SQL command:
   `@ $WL_COMMERCE_HOME/db/`*`database-type`*`/upgrade-wlcs-to-320.sql.`

Note: Enter this command only once and only after you have modified all rows that contain data exceeding new length requirements.

When the command successfully completes updating tables, it prints the following message:

```
************  SCRIPT HAS FINISHED EXECUTING   ************

******  PLEASE REVIEW UPDATE-TO-320.LOG  FILE  *****
```

Verify the Upgrade

After you upgrade the schema for each server that you are using, verify the upgrade by starting the server and Administration Tool and testing the application. For example, if you use both Commerce services and WebLogic Personalization Server, open the Administration Tool to verify that the users and groups you upgraded are available under User Administration, and all items and categories that you upgraded are available under Catalog Administration. Then access the server through a Web browser to verify that data transferred successfully.

Starting the Server

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

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

- Click Start → Programs → Commerce services 3.2 → Start Commerce Server.
- From a command prompt, enter the following command:
  `%WL_COMMERCE_HOME%\StartCommerce.bat`

Remove Temporary Tables

Note: Do not complete this step until you have successfully upgraded the schema for **both** servers (if you use both servers) to Release 3.2 **and** started the application and verified the data migration.

After you have verified that WebLogic Commerce Server and WebLogic Personalization Server function properly with the imported data, remove the temporary `BEA_`*`table-name`* tables by running the following SQL command:
`@ $WL_COMMERCE_HOME/db/`*`database-type`*`/drop_bea_tables.sql`

When the command successfully completes removing `BEA_` tables, it prints the following message:

```
************  SCRIPT HAS FINISHED EXECUTING   *************
******  PLEASE REVIEW  DROP_BEA_TABLES.LOG  FILE  *****
```

Verifying the Migration to 3.2

In a test environment, run a typical selection of common user and administrator tasks to be sure your system was migrated correctly. Suggested tasks include the following:

- Go to the example portal and be sure you can log in. See "Overview of the Personalization Tour" in the *Personalization Server Tour* documentation.
- Go to the example Commerce Server application and make sure you can both view items and add them to a shopping cart. See "Registered User Buys a Product" in the *Tour of the JSP Templates* documentation.
- Log into the administration tools and view properties for several users. See "Working with User Profiles" in the *Personalization Server Tour* documentation.

Refer to the appropriate section of the referenced documentation and this guide if you encounter errors.

What's Next: Getting Started With 3.2

Use the online documentation site to locate the documentation you need for additional learning.

http://edocs.bea.com/wlcs/docs32/index.htm

For an overview of how the products work together, see the *Product Family Overview* at http://edocs.bea.com/wlcs/docs32/intro/index.htm.

5 Migrating WebLogic Personalization Server From Version 2.0.1 to Version 3.1

This chapter describes the changes between WebLogic Personalization Server 2.0.1 and WebLogic Personalization Server 3.1. It includes specific information for migrating existing code to WebLogic Personalization Server 3.1.

Note: Both the Commerce services and WebLogic Personalization Server functionality now reside in a unified Java package hierarchy located at com.beasys.commerce.

Note: This covers Personalization Server migration only. BEA is addressing a strategy for migrating WebLogic Commerce Server 2.01.

Note: You typically must migrate from one release to the next consecutive one. However, there is one exception. To migrate WebLogic Personalization Server from 2.0.1 to 3.2, see "2.0.1 to 3.2: WebLogic Personalization Server" on page 4-53, within "Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2."

This section includes the following topics:

- Migration Checklist
- What's New

- Migrating Code
- Migrating Data
- Configuring Systemwide Settings
- Verifying the Migration
- What's Next

Migration Checklist for 2.0.1 to 3.1

Follow the steps in this section to migrate your system to the current release.

1. Be sure you've already reviewed the information in Chapter 1, "Planning Your Migration," beginning on page 1-1.
2. The bulk of the code migration consists of your reviewing the new and changed features, locating where in your code you use or depend on any of them, and making the appropriate changes. Review "What's New in 3.1" on page 5-3 so you know the main types of issues involved; the subsequent migration sections provide more detail and instructions.
3. Follow the steps in "Migrating Code From 2.0.1 to 3.1" on page 5-7 to make the necessary changes to your code.
4. Follow the steps in "Migrating Data From 2.0.1 to 3.1" on page 5-35 to migrate WebLogic Personalization Server database schemas from 2.0.1 to 3.1.1

   Note: To upgrade from 2.0.1 to 3.2, see "Migrating Data" on page 5-85 of *Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2*.
5. Read "New Custom Tags and Migrating Custom Tags" on page 5-20 to learn details about the new custom tags, and to make the necessary changes to tags that have been changed.
6. Follow the steps in "Verifying the Migration to 3.1" on page 5-37 to ensure that your migration was completed successfully.
7. Read "What's Next: Getting Started With 3.1" on page 5-38 for suggestions on next steps and additional information.

What's New in 3.1

Note: Be sure to also read the "What's New" page on the e-docs.bea.com Web site for a preview of the new features in this release: http://e-docs.bea.com/wlcs/docs31/interm/whatsnew.htm

This section briefly lists the notable changes to this release. Instructions for migrating your system to accommodate these changes are in "Migrating Code" on page 5-71.

The information is organized as follows:

- New Custom Tags
- Navigating With Flow Manager
- Changes to the Personalization Advisor
- Changes to the Rules Editor in Release 3.1
- Changes to Content Management

New Custom Tags

This topic includes the following changes; see "New Custom Tags and Migrating Custom Tags" on page 5-87 for details and migration information.

Five new tags were introduced in WebLogic Personalization Server Release 3.1.

- <ps:getPropertyNames>
- <ps:getPropertySetNames>
- <i18n:localize>
- <i18n:getMessage>
- <wl:repeat>

Considerable changes have been made to the existing custom tags, as well. See "Migrating Existing JSP Tags" on page 5-89 for more information.

Navigating With Flow Manager

This topic includes the following changes; see “Migrating to Flow Manager” on page 5-71 for details and migration information.

The Flow Manager is a servlet implementation that allows the hot deployment of applications within the WebLogic Application Server. Flow Manager also adds flexibility to navigation through the system—it moves navigation information off the JSP page and into a single point of control. Using a destination determiner and a destination handler, the Flow Manager dynamically determines a destination for a given page request and dynamically handles it.

- Deprecated Service Managers – In WebLogic Personalization Server 3.1, all of the functionality of the JSP Service Manager and the Portal Service Manager has been ported to the new Flow Manager. The JSP Service Manager and the Portal Service Manager have been deprecated.
- Hot Deployment – The Flow Manager is a servlet implementation that allows the hot deployment of applications within the WebLogic Application Server. Registering a new portal or a new application no longer requires restarting the server, as it did in WebLogic Personalization Server 2.0.1. Instead of registering servlets in the `weblogic.properties` file, the Flow Manager relies on a property set to obtain information about a specific application or portal. You simply create a new instance of a property set to hold the equivalent parameters that were in the properties file. Default values are supplied during property set creation. Any changes become visible according to a configurable refresh setting in the property set.
- Dynamic Flow Determination and Handling – Flow Manager also provides the basic infrastructure to support the new Webflow functionality. Webflow dynamically determines a destination for a given page request and dynamically handles it. Using a destination determiner and a destination handler, the Flow Manager moves navigation information off the JSP page and into a single point of control.
- Property Set Usage – A new class of property sets, “Application Initialization Property Sets” has been added to the Property Set Management Administration Tools. These are the property sets used by the Flow Manager in support of portal (`_DEFAULT_PORTAL_INIT`) and non-portal-based (`_DEFAULT_APP_INIT`) personalized applications. Three new properties have been added to support the Flow Manager.

Changes to the Personalization Advisor

This topic includes the following changes; see "Migrating the Personalization Advisor" on page 5-77 for details and migration information.

For WebLogic Personalization Server 3.1, the Personalization Advisor has been renamed to Advisor and has undergone some API changes. However, its functionality remains the same. The Advisor has been improved to provide better error reporting and to make use of the unified logging facility provided by Commerce services 3.1.

- JSP Tags Ported to Use the New Advisor – The three `pz` library tags (`pz:div`, `pz:contentQuery`, and `pz:contentSelector`) have been changed to use the new Advisor Session Bean. However, the tag usage remains the same. For more information, see the JSP Tag Library Reference in the *WebLogic Personalization Server Developer's Guide* (in the release 3.1 documentation set).
- Deprecated Personalization Advisor Classes – All of the Java classes for the Personalization Advisor released in WebLogic Personalization Server 2.0.1 have been deprecated.
- Changes in Advisor APIs – Changes were made while porting the WebLogic Personalization Server 2.0.1 Personalization Advisor interface to the new Advisor interfac. The Personalization Advisor `pzTechnique` parameter is not supported in the new Advisor implementation; and the `createRequestTemplate` method parameters have been simplified to use a single string lookup name for the advice request, instead of a fully qualified class name.
- Terminology Change: Agents Changed to Advislets – The three WebLogic Personalization Server 2.0.1 Personalization Agents have been renamed and repackaged to advislets. The following table defines the mapping between the WebLogic Personalization Server 2.0.1 Agent Java classes to the WebLogic Personalization Server 3.0 advislet Java classes.

Changes to the Rules Editor in Release 3.1

This topic includes the following changes; see "Migrating the Rules Editor" on page 5-80 for details and migration information.

The WebLogic Personalization Server provides rule sets that include a set of classifier and content selector rules. These rule sets act as containers for rules that match personalized content with users.

- Relationship Between Rules and Property Sets – In previous releases, the rule sets (also called rule sheets) were associated with property sets that defined the attributes available for user and group profiles. Once defined, this relationship between rules and property sets could not be undone.
- The Use of AND or OR to Connect Expressions – The Rules Editor now allows the use of "and" or "or" to connect expressions that contain comparators.
- Change the Word 'Rule Sheet' to 'Rule Set' – For consistency, an effort has been made to change the word "rule sheet" to "rule set" or `ruleSet` in all cases. However, the following legacy code continues to use Rulesheet: `jdbc://com.beasys.commerce.axiom.reasoning.rules.RulesheetDefinitionHome`

Changes to Content Management

This topic includes the following changes; see "Migrating Content Management" on page 5-81 for details and migration information.

- New Features in <cm:select> and <cm:selectById> Tags – To retrieve Content or Documents, use a ContentManager or DocumentManager with `<cm:select>` or `<cm:selectById>`. The default DocumentManager is deployed at `com.beasys.commerce.axiom.document.DocumentManager`.
- Document Schema EJB Deployment Descriptor – Two EJB variables have been removed and five EJB variables have been added.
- DocumentManager EJB Deployment Descriptor – All the EJB variables have been removed and six EJB variables have been added.
- Document EJB Deployment Descriptor (Deprecated) – The Document EJB has been deprecated and should not be used. Use the DocumentManager EJB instead. To support legacy code, the Document EJB has been upgraded with removal of two EJB variables and the addition of four new EJB variables.
- Changes to Object Interfaces – The `ConfigurableEntity`, `Content`, `Document`, `User` and `Group` interfaces no longer extend EJBObject. Instead,

those interfaces are code-identical to the original 2.0.1 versions (same method signatures).

- Changes to the BulkLoader – The BulkLoader now accepts an `-encoding <enc>` and `-schemaName` option. For more information, see "Command Line Usage" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

Migrating Code From 2.0.1 to 3.1

Use the information in this section to migrate the code, including JSP tags. The information in this section is organized as follows:

- Migrating to Flow Manager
- Migrating the Personalization Advisor
- Migrating the Rules Editor
- Migrating Content Management
- Migrating the Personalization Advisor

Migrating to Flow Manager

This section covers the features in Flow Manager, and describes the steps you must take in order to use it. For more information, see "Flow Manager" in the Foundation chapter in the *WebLogic Personalization Server Developer's Guide* (in the release 3.1 documentation set).

The Flow Manager is a servlet implementation that allows the hot deployment of applications within the WebLogic Application Server. Flow Manager also adds flexibility to navigation through the system—it moves navigation information off the JSP page and into a single point of control. Using a destination determiner and a destination handler, the Flow Manager dynamically determines a destination for a given page request and dynamically handles it.

This topic includes the following sections:

- About Flow Manager
- Migrating to the Flow Manager

About Flow Manager

This section covers features of Flow Manager.

- Deprecated Service Managers
- Hot Deployment
- Dynamic Flow Determination and Handling
- Property Set Usage
- Deprecated Service Managers

Deprecated Service Managers

In WebLogic Personalization Server 3.1, all of the functionality of the JSP Service Manager and the Portal Service Manager has been ported to the new Flow Manager. The JSP Service Manager and the Portal Service Manager have been deprecated.

Hot Deployment

The Flow Manager is a servlet implementation that allows the hot deployment of applications within the WebLogic Application Server.

Registering a new portal or a new application no longer requires restarting the server, as it did in WebLogic Personalization Server 2.0.1. Instead of registering servlets in the `weblogic.properties` file, the Flow Manager relies on a property set to obtain information about a specific application or portal. You simply create a new instance of a property set to hold the equivalent parameters that were in the properties file. Default values are supplied during property set creation. Any changes become visible according to a configurable refresh setting in the property set.

Note: In Release 3.5, the direction of the product will be to deploy one portal per Web application and use the application deployment features enabled through the WLS console, instead of the hot deployment portal model. For more information, see "All Pages Must Be in a Web Application" in Chapter 1, "Migrating WebLogic Commerce Server to Version 3.5."

Dynamic Flow Determination and Handling

Flow Manager also provides the basic infrastructure to support the new Webflow functionality. Webflow dynamically determines a destination for a given page request and dynamically handles it. Using a destination determiner and a destination handler, the Flow Manager moves navigation information off the JSP page and into a single point of control.

The old service managers relied on a hidden form field in the current page to determine where an HTTP request should be routed:

```
<input type="hidden" name="<%=DESTINATION_TAG%>"
value="<%=PortalJspBase.getRequestURI(request)%>">
```

This scheme required destination (or routing) information to be distributed across the JSP/HTML pages. While this works fine, it can be cumbersome to modify if destination values need to change.

The Flow Manager, on the other hand, allows the determination of page routing to be centralized on the server based on an application's needs.

Backward Compatibility

For backward compatibility, default implementations of the destination determiner and the destination handler are provided which support destination information being passed via the DESTINATION_TAG mentioned above. These implementations are:

`com.beasys.commerce.portal.flow.PortalDestinationDeterminer`
and
`com.beasys.commerce.foundation.flow.ServletDestinationHandler`

Also, for non-portal-based personalized applications, the following default implementations may be used:

`com.beasys.commerce.foundation.flow.jsp.DefaultDestinationDeterminer`
and
`com.beasys.commerce.foundation.flow.ServletDestinationHandler`

Property Set Usage

A new class of property sets, “Application Initialization Property Sets” has been added to the Property Set Management Administration Tools. These are the property sets used by the Flow Manager in support of portal (`_DEFAULT_PORTAL_INIT`) and non-portal-based (`_DEFAULT_APP_INIT`) personalized applications.

Three new properties have been added to support the Flow Manager:

- **destinationdeterminer** Property

  The destination determiner is responsible for evaluating an HTTP request and determining which servlet to route it to.

  The value provided for this property should be the name of a class that implements the `com.beasys.commerce.foundation.flow.DestinationDeterminer` interface. If appropriate, use a default implementation provided by WebLogic Personalization Server or Commerce services. Otherwise, develop your own implementation according to the needs of your application.

- **destinationhandler** Property

  Given a destination route, the destination handler is responsible for invoking the requested processing.

  The value provided for this property should be the name of a class that implements the `com.beasys.commerce.foundation.flow.DestinationHandler` interface. If appropriate, use a default implementation provided by WebLogic Personalization Server or Commerce services. Otherwise, develop your own implementation according to the needs of your application.

- **ttl (time-to-live)** Property

  ttl, which stands for time-to-live, represents how often (in milliseconds) the Flow Manager reloads the `_APPLICATION_INIT` property set from the database. This allows changes that you make to the `_APPLICATION_INIT` property set to be visible while the application or portal is running.

Note: To force immediate reloading of the property set, append the "flowReset" argument to your URL, like this:
`http://localhost:7001/application/exampleportal?flowReset=true`

Migrating to the Flow Manager

To migrate your portal or non-portal application to use the Flow Manager, do the following:

To create a new property set:

1. Open the Administration Tools Home page. Click the Property Set Management icon to open the Property Set Management screen.
2. From the main Property Set Management screen, click Create.
3. Name the new property set you are creating (100 character maximum). The name of the property set should be the same as the name you used to create the portal, or the name you will use to access the application.
4. Enter a description of the property set (255 character maximum).
5. From the Copy Properties From drop-down list, select
   `APPLICATION_INIT._DEFAULT_PORTAL_INIT` (for a portal)
   or
   `APPLICATION_INIT._DEFAULT_APP_INIT` (for a non-portal application).
6. From the Property Set Type drop-down list, select Application Init.
7. Click the Create button.
8. At the top of the page, in red, you will see the message "Property Set creation was successful." (Or, you will see an error message indicating why the property set was not created.)
9. Click Back to return to the main Property Set Management screen.

To set parameters for your portal or application:

1. From the Property Set Management Home page, under the Application Initialization Property Sets heading, click the name of the property set you just created.
2. A Property Set page comes up, allowing you to set parameters.
3. **(Note:** For non-portal applications, skip this step.) To edit the portal name, click the Edit button to the right of the "portal name" property. Change the default value from `UNKNOWN` to the name of your portal, as you created it in Portal Management.
4. Edit the `destinationdeterminer` property. Either accept the default, or edit to provide your own implementation of these classes.
5. Edit the `destinationhandler` property. Either accept the default, or edit to provide your own implementation of these classes.

6. Customize any other properties you choose. For information about customizing properties in portals, see Creating and Managing Portals in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set) and Building a Custom Portal Step-by-Step in the*WebLogic Personalization Server Developer's Guide* (in the release 3.1 documentation set).
7. When you have finished setting properties, click the Finished button at the bottom of the page.

Note: In WebLogic Personalization Server 2.0.1, you registered servlets in the `weblogic.properties` file. This is not required for WebLogic Personalization Server 3.1. You have the option to remove them, but it is not required. The WebLogic Personalization Server will ignore them.

Accessing Your Application via the Flow Manager

The exact URL you use depends upon whether or not you have deployed your application as a Web application. WebLogic Personalization Server 3.1 includes sample configurations for both a Web application/Web archive deployment and a non-Web application configuration. For more information, see the chapter "Using the Catalog Application in a Portal" in the *WebLogic Personalization Server Developer's Guide* (in the release 3.1 documentation set).

Migrating the Personalization Advisor

For WebLogic Personalization Server 3.1, the Personalization Advisor has been renamed to Advisor and has undergone some API changes. However, its functionality remains the same. The Advisor has been improved to provide better error reporting and to make use of the unified logging facility provided by Commerce services 3.1.

Update your implementation as appropriate based on the following changes.

This topic includes the following sections:

- JSP Tags Ported to Use the New Advisor – The three `pz` library tags (`pz:div`, `pz:contentQuery`, and `pz:contentSelector`) have been changed to use the new Advisor Session Bean. However, the tag usage remains the same. For more information, see the JSP Tag Library Reference in the WebLogic Personalization Server Developer's Guide (in the release 3.1 documentation set).

- Deprecated Personalization Advisor Classes – All of the Java classes for the Personalization Advisor released in WebLogic Personalization Server 2.0.1 have been deprecated.
- Changes in Advisor APIs – Changes were made while porting the WebLogic Personalization Server 2.0.1 Personalization Advisor interface to the new Advisor interfac. The Personalization Advisor `pzTechnique` parameter is not supported in the new Advisor implementation; and the `createRequestTemplate` method parameters have been simplified to use a single string lookup name for the advice request, instead of a fully qualified class name.
- Terminology Change: Agents Changed to Advislets – The three WebLogic Personalization Server 2.0.1 Personalization Agents have been renamed and repackaged to advislets. The following table defines the mapping between the WebLogic Personalization Server 2.0.1 Agent Java classes to the WebLogic Personalization Server 3.0 advislet Java classes.

JSP Tags Ported to Use the New Advisor

The three `pz` library tags (`pz:div`, `pz:contentQuery`, and `pz:contentSelector`) have been changed to use the new Advisor Session Bean. However, the tag usage remains the same. For more information, see the JSP Tag Library Reference in the *WebLogic Personalization Server Developer's Guide* (in the release 3.1 documentation set).

To use the `<pz:div>` and `<pz:contentSelector>` tags, you are no longer required to insert the following JSP directive into your JSP code:

```
<%@ page extends="com.beasys.commerce.axiom.p13n.jsp.P13NJspBase"
%>
```

However if it is already in your code, you do not need to remove it.

Deprecated Personalization Advisor Classes

All of the Java classes for the Personalization Advisor released in WebLogic Personalization Server 2.0.1 have been deprecated. This includes all of the Java classes in the following Java packages:

```
com.beasys.commerce.axiom.p13n.advisor
com.beasys.commerce.axiom.p13n.agents
```

In WebLogic Personalization Server 3.1, these deprecated classes are replaced by new Advisor Java packages. They include:
`com.beasys.commerce.axiom.advisor`
`com.beasys.commerce.axiom.advislets`

The Personalization Advisor Bean has been replaced by the new Advisor Bean.

This change only affects the case when the Advisor API is used directly and is transparent to JSP tag users.

Changes in Advisor APIs

The changes made while porting the WebLogic Personalization Server 2.0.1 Personalization Advisor interface to the new Advisor interface are as follows:

- The Personalization Advisor `pzTechnique` parameter is not supported in the new Advisor implementation.
- The `createRequestTemplate` method parameters have been simplified to use a single string lookup name for the advice request, instead of a fully qualified class name. The three advice request lookup names supported for WebLogic Personalization Server 3.1 are `ClassificationAdviceRequest`, `ContentSelectorAdviceRequest`, and `ContentQueryAdviceRequest`.

The following example shows the difference in the `createRequestTemplate` method between the Personalization Advisor and the Advisor.

Personalization Advisor Interface

```
public AdviceRequest createRequestTemplate(
    String adviceRequestClassName,
    String pzTechnique)
    throws IllegalArgumentException,
           PersonalizationAdvisorException,
           RemoteException;
```

Advisor Interface

```
public AdviceRequest createRequestTemplate(
    String theKindOfRequest)
    throws IllegalArgumentException,
           AdvisorException,
           RemoteException;
```

Terminology Change: Agents Changed to Advislets

The three WebLogic Personalization Server 2.0.1 Personalization Agents have been renamed and repackaged to advislets. The following table defines the mapping between the WebLogic Personalization Server 2.0.1 Agent Java classes to the WebLogic Personalization Server 3.0 advislet Java classes.

2.0 Agent class	com.beasys.commerce.axiom.p13n.agents.ClassificationAgentImpl
3.1 Advislet class	com.beasys.commerce.axiom.advisor.advislets.ClassificationAdvisletImpl
2.0 Agent class	com.beasys.commerce.axiom.p13n.agents. ContentSelectorAgentImpl
3.1 Advislet class	com.beasys.commerce.axiom.advisor.advislets.ContentSelectorAdvisletImpl
2.0 Agent class	com.beasys.commerce.axiom.p13n.agents. ContentQueryAgentImpl
3.1 Advislet class	com.beasys.commerce.axiom.advisor.advislets.ContentQueryAdvisletImpl

Migrating the Rules Editor

The WebLogic Personalization Server provides rule sets that include a set of classifier and content selector rules. These rule sets act as containers for rules that match personalized content with users.

Update your system as appropriate based on the information in the following sections:

- Relationship Between Rules and Property Sets – In previous releases, the rule sets (also called rule sheets) were associated with property sets that defined the attributes available for user and group profiles. Once defined, this relationship between rules and property sets could not be undone.
- The Use of AND or OR to Connect Expressions – The Rules Editor now allows the use of “and” or “or” to connect expressions that contain comparators.
- Change the Word ‘Rule Sheet’ to ‘Rule Set’ – For consistency, an effort has been made to change the word “rule sheet” to “rule set” or `ruleSet` in all cases. However, the following legacy code continues to use Rulesheet: `jdbc://com.beasys.commerce.axiom.reasoning.rules.RulesheetDefinitionHome`

For more information, see "Creating and Managing Rules" in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

Relationship Between Rules and Property Sets

In previous releases, the rule sets (also called rule sheets) were associated with property sets that defined the attributes available for user and group profiles. Once defined, this relationship between rules and property sets could not be undone.

In the current WebLogic Personalization Server 3.1 release, there is no longer an association between a rule set and a property set. Rules within a rule set may refer to any properties.

The Use of AND or OR to Connect Expressions

The Rules Editor now allows the use of "and" or "or" to connect expressions that contain comparators.

Change the Word 'Rule Sheet' to 'Rule Set'

For consistency, an effort has been made to change the word "rule sheet" to "rule set" or `ruleSet` in all cases. However, the following legacy code continues to use Rulesheet:
`jdbc://com.beasys.commerce.axiom.reasoning.rules.RulesheetDefinitionHome`

Migrating Content Management

Make changes to your system as appropriate based on the information in the following sections:

- New Features in <cm:select> and <cm:selectById> Tags – To retrieve Content or Documents, use a ContentManager or DocumentManager with `<cm:select>` or `<cm:selectById>`. The default DocumentManager is deployed at `com.beasys.commerce.axiom.document.DocumentManager`.
- Changes to EJB Deployment Descriptors

- Changes to Object Interfaces – The `ConfigurableEntity, Content, Document, User` and `Group` interfaces no longer extend EJBObject. Instead, those interfaces are code-identical to the original 2.0.1 versions (same method signatures).
- Changes to the BulkLoader – The BulkLoader now accepts a `-encoding <enc>` and `-schemaName` option. For more information, see "Command Line Usage" in the chapter Creating and Managing Content in the WebLogic Personalization Server User's Guide (in the release 3.1 documentation set).

New Features in <cm:select> and <cm:selectById> Tags

To retrieve Content or Documents, use a ContentManager or DocumentManager with `<cm:select>` or `<cm:selectById>`. The default DocumentManager is deployed at `com.beasys.commerce.axiom.document.DocumentManager`. For more information, see "Configuring WebLogic Commerce Properties" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

The `<cm:select>` and `<cm:selectById>` tags now support a session-based, per-user Content cache for content searches. For more information, see "Content Cache" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

The Content Manager now supports non-EJB context objects. The `<cm:select>` and `<cm:selectById>` tags support an optional `readOnly` parameter. For more information, see "readOnly Content Tag" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

Changes to EJB Deployment Descriptors

Deployment descriptors handle the configuration for the Content Manager. This section describes the changes to the deployment descriptors:

- Document Schema EJB Deployment Descriptor
- DocumentManager EJB Deployment Descriptor – All the EJB variables have been removed and six EJB variables have been added.
- Document EJB Deployment Descriptor (Deprecated) – The Document EJB has been deprecated and should not be used. Use the DocumentManager EJB

instead. To support legacy code, the Document EJB has been upgraded with removal of two EJB variables and the addition of four new EJB variables.

Document Schema EJB Deployment Descriptor

Two EJB variables have been removed:
`SmartConnectionPoolClass`
`SmartBMP_URL`

Five EJB variables have been added:
`UseDataSource`
`DocPoolURL`
`DocPoolDriver`
`jdbc/docPool`
`jdbc/commercePool`

One EJB variable remains the same:
`SmartBMPUpdate`

For more information, see "Configuring the Document Schema EJB Deployment Descriptor" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

DocumentManager EJB Deployment Descriptor

All the EJB variables have been removed:
`UseDefaultHomeNames`
`ContentHome`
`SchemaHome`

Six EJB variables have been added:
`PropertyCase`
`jdbc/docPool`
`ejb/ContentHome`
`ejb/SchemaHome`
`UseDataSource`
`DocPoolURL`
`DocPoolDriver`

For more information, see "Configuring the DocumentManager EJB Deployment Descriptor" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

Document EJB Deployment Descriptor (Deprecated)

Note: The Document EJB has been deprecated and should not be used. Use the DocumentManager EJB instead.

To support legacy code, the Document EJB has been upgraded as follows:

Two EJB variables have been removed:
`SmartConnectionPoolClass`
`SmartBMP_URL`

Four EJB variables have been added:
`UseDataSource`
`DocPoolURL`
`DocPoolDriver`
`jdbc/docPool`

Two EJB variables remain the same:
`SmartBMPUpdate`
`Propertycase`

- *`SmartBMPUpdate`*: Set to `false`.
- *`UseDataSource`*: Controls whether `jdbc/docPool` (`true`) or `DocPoolURL` (`false`) is used to get connections. Defaults to `true`.
- *`DocPoolURL`*: Specifies the JDBC URL to the document JDBC connection to use (if `UseDataSource` is `false`). Should point to a connection pool.
  For example: `jdbc:weblogic:pool:docPool`
- *`DocPoolDriver`*: Specifies the JDBC driver class to use to connect to the `DocPoolURL`. This is optional. If not specified, the EJB will try to determine the appropriate JDBC driver class from the `DocPoolURL`.
- *`jdbc/docPool`*: A Data Source reference to the document JDBC connection pool. This should correspond to the Data Source attached to the WebLogic connection pool that uses the document reference implementation JDBC driver.
- *`PropertyCase`*: This sets how the `DocumentImpl` modifies incoming property names. If this is *lower*, all property names are converted to lowercase. If this is *upper*, all property names are converted to uppercase. If this is anything else or not specified, property names are not modified. Use *lower* or *upper* if the `SmartBMP` class expects everything in a certain case (for example, the Documentum `SmartBMP` expects everything in lowercase). For the document reference implementation, do not specify the *`PropertyCase`*.

Other `SmartBMP` classes for other document management systems will possibly require more and/or different EJB environment variables.

Changes to Object Interfaces

The `ConfigurableEntity`, `Content`, `Document`, `User` and `Group` interfaces no longer extend EJBObject. Instead, those interfaces are code-identical to the original 2.0.1 versions (same method signatures).

The interfaces `ConfigurableEntityRemote`, `ContentRemote`, `DocumentRemote`, `UserRemote` and `GroupRemote` extend both EJBObject and their respective non-EJBObject interfaces.

For more information, see "Object Interfaces" in the chapter "Creating and Managing Content" in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

Changes to the BulkLoader

The BulkLoader now accepts a `-encoding <enc>` and `-schemaName` option. For more information, see "Command Line Usage" in the chapter Creating and Managing Content in the *WebLogic Personalization Server User's Guide* (in the release 3.1 documentation set).

New Custom Tags and Migrating Custom Tags

This section covers the new JSP tags in this release, as well as the changes to JSP tags and what you need to migrate your system to accommodate those changes.

- Migrating Existing JSP Tags
- Additional Notes About Existing JSP Tags
- New JSP Tags in Release 3.1

Migrating Existing JSP Tags

This section covers the *changed* tags in this release; see "New JSP Tags in Release 3.1" on page 5-34 for *new* tags and migration information.

Update your use of these tags as indicated by the changes described in this section, and review code that references these tags to be sure you have addressed all dependencies.

The tag libraries have been updated in WebLogic Personalization Server version 3.1 to comply with the JSP 1.1 Specification. If you are upgrading from WebLogic Personalization Server 2.0.1, you can continue to use your existing code with WebLogic Personalization Server 3.1. However, *future releases will no longer be backward compatible*, so you will need to migrate to the new tags if you intend to continue to use your legacy code with the latest WebLogic Personalization Server releases.

The WebLogic Personalization Server 3.1 documentation has been revised to reflect the changes to the tag libraries. Until you migrate to the new tags, you can continue to use the WebLogic Personalization Server 2.0 *JSP Tag Reference* located at http://e-docs.bea.com/wlcs/docs20/p13ndev/jsptags.htm.

New JSP 1.1 Naming Conventions

Beginning with WebLogic Personalization Server version 3.1, all tags use the JSP 1.1 naming conventions. Old style tags that were used in previous WebLogic Personalization Server releases have been changed to reflect the new camel case naming conventions.

For example, the old-style tag `<um:getgroupnamesforusers>` is now `<um:getGroupNamesForUsers>`.

Old tag names can still be used in the WebLogic Personalization Server 3.1 release. However, *old style tag names will not be supported in future releases of* WebLogic Personalization Server.

Note: Each time you use a deprecated tag, a message is logged to WebLogic Server. To turn off the deprecation messages, add the following property to weblogiccommerce.properties:
`commerce.log.display.deprecated=false`

For consistency, the Portal Management tags `<pt:*>` have a new `esp:` prefix. For example, the old-style tag `<pt:eval>` is now called `<esp:eval>`, and the old `<pt:portalmanager>` is now `<esp:portalManager>`. When you change to the new prefix, you will need to update each Portal Management tag invocation in the page to use the new prefix.

Note: The `es:` prefix stands for e-commerce services.
The `esp:` prefix stands for e-commerce services portal.
The `pz:` prefix stands for personalization.

Changes to Tag Attributes

- For the Content Management `<cm:printDoc>` tag, a new attribute, `baseHref`, has been added. This attribute provides the URL of the document's BASE HREF.
- For the User Management `<um:*>` tags, the `resultId` attribute has been changed to `result`, and is now an Integer instead of an int. Usage and functionality remain the same.
- For the User Management tags `<um:getProperty>` and `<um:setProperty>`, the `usecache` attribute has been dropped.
- For the WebLogic Personalization Server Utility tags `<es:isNull>` and `<es:notNull>`, the `id` attribute has been changed to `item`.
- For the WebLogic Personalization Server Utility tag `<es:preparedStatement>`, the `pool` attribute has been dropped (see "Note 4: <es:preparedStatement>" on page 5-101) and a new attribute, `transactionIsolationLevel`, has been added.

Tag Attributes Require Camel Casing

All of the tag attributes used in previous WebLogic Personalization Server releases already use the camel-case convention, with a few exceptions. The tags that do not already use camel-cased attributes are the three Advisor tags (formerly called Personalization Advisor) `<pz:*>`, and the single WebLogic utility `<wl:process>`.

Table 5-1 lists the attributes that you will need to camel case. Note that all of these attributes are optional, so it is possible that you did not use them in your existing code.

Table 5-1 Camel-Cased Attributes

Tag	Attribute
`<pz:div>`	`ruleSet`
`<pz:contentQuery>`	`sortBy` `contentHome`

Table 5-1 Camel-Cased Attributes

Tag	Attribute
<pz:contentSelector>	ruleSet sortBy contentHome
<wl:process>	notName notValue

New Library Descriptors

Any JSP migrating from old-style tags to new-style tags will need to point to new library descriptors.

- For Portal Management <pt:*> tags, change "lib/esportal.jar" to "esp.tld". (Also, change prefix="pt" to prefix= "esp". Update each invocation of a Portal Management tag on the page to use the "esp" prefix.)
- For User Management <um:*> tags, change "lib/um_tags.jar" to "um.tld".
- For Personalization Utilities <es:*> tags, change "lib/esjsp.jar" to "es.tld".
- For the WebLogic Utility <wl:process> tag, change "lib/wljsp.jar" to "weblogic.tld".

For example:

In the JSP page, <%@ taglib uri="lib/um_tags.jar" prefix="um" %> would change to <%@ taglib uri="um.tld" prefix="um" %>.

Note: The Personalization Advisor is now simply called the Advisor. The Advisor <pz:*> tags already use taglib uri="pz.tld", so these do not need to be changed.

The Content Management <cm:*> tags already use taglib uri="cm.tld", so these do not need to be changed.

Global Changes

Tags no longer return primitive types, they only return objects. For example, `<es:counter>` used to return an int, and now it returns an Integer object.

Any tags (`es`, `um`, `wl`, etc.) with a `<jsp:include page=.../>` in their body must be replaced with their scriptlet equivalent. (See Section 5.4.5 of the JSP 1.1 Specification.)

Old Usage:

```
<es:notNull item="renderer">
  <jsp:include page="<%=reconcileFile(request, renderer)%>"/>
</es:notNull>
```

New Usage:

```
<% if (renderer != null) { %>
  <jsp:include page="<%=reconcileFile(request, renderer)%>"/>
<% } %>
```

Tag Migration Roadmap

Table 5-2 maps the old tag names to the new JSP 1.1 camel-cased tag names. In addition, changes made to the tags in the WebLogic Personalization Server 3.1 release are noted in the Change column.

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
Advisor	`<pz:contentquery>`	Camel case Attribute `sortby = sortBy` Attribute `contenthome = contentHome` It is no longer necessary to extend the JSP. See below - **Note 1: <pz:> tags**.	`<pz:contentQuery>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<pz:contentselector>`	Camel case Attribute `ruleset = ruleSet` Attribute `sortby = sortBy` Attribute `contenthome = contentHome`	`<pz:contentSelector>`
	`<pz:div>`	`ruleset = ruleSet` It is no longer necessary to extend the JSP. See below - **Note 1: <pz:> tags**.	`<pz:div>`
Content Mngmt	---	**New**	`<cm:getProperty>`
	`<cm:printproperty>`	Camel case	`<cm:printProperty>`
	`<cm:printdoc>`	Camel case New attribute: `baseHref`	`<cm:printDoc>`
	`<cm:select>`	No change	`<cm:select>`
	`<cm:selectbyid>`	Camel case	`<cm:selectById>`
Flow Manager	---	**New**	`<fm:getApplicationURI>`
	---	**New**	`<fm:getCachedAttribute>`
	---	**New**	`<fm:setCachedAttribute>`
	---	**New**	`<fm:removeCachedAttribute>`
	---	**New**	`<fm:getSessionAttribute>`
	---	**New**	`<fm:setSessionAttribute>`
	---	**New**	`<fm:removeSessionAttribute>`
I18N	---	**New**	`<i18n:initialize>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	---	**New**	`<i18n:getMessage>`
Property Set	---	**New**	`<ps:getPropertyName>`
	---	**New**	`<ps:setPropertyName>`
Portal	`<pt:eval>`	`taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:eval>`
	`<pt:get>`	`taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:get>`
	`<pt:getgroupsforportal>`	Camel case Change preface `pt:` to `esp:` `taglib uri="esp.tld"`	`<esp:getGroupsForPortal>`
	`<pt:monitorsession>`	Camel case `taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:monitorSession>`
	`<pt:portalmanager>`	Camel case `taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:portalManager>`
	`<pt:portletmanager>`	Camel case `taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:portletManager>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<pt:props>`	`taglib uri="esp.tld"` Change preface `pt:` to `esp:`	`<esp:props>`
User/ Profile	`<um:getprofile>`	Camel case `taglib uri="um.tld"`	`<um:getProfile>`
	`<um:getproperty>`	Camel case `taglib uri="um.tldv"`	`<um:getProperty>`
	`<um:getpropertyasstring>`	Camel case `taglib uri="um.tld"`	`<um:getPropertyAsString>`
	`<um:removeproperty>`	Camel case `taglib uri="um.tld"`	`<um:removeProperty>`
	`<um:setproperty>`	Camel case `taglib uri="um.tld"`	`<um:setProperty>`
User/ Group	`<um:addgrouptogroup>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:addGroupToGroup>`
	`<um:addusertogroup>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:addUserToGroup>`
	`<um:changegroupname>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:changeGroupName>`
	`<um:creategroup>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:createGroup>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<um:createuser>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:createUser>`
	`<um:getchildgroupnames>` *(previously undocumented)*	Camel case `taglib uri="um.tld"`	`<um:getChildGroupNames>`
	`<um:getchildgroups>`	Camel case `taglib uri="um.tld"`	`<um:getChildGroups>`
	`<um:getgroupnamesforuser>`	Camel case `taglib uri="um.tld"`	`<um:getGroupNamesForUser>`
	`<um:getparentgroupname>`	Camel case `taglib uri="um.tld"`	`<um:getParentGroupName>`
	`<um:gettoplevelgroups>`	Camel case `taglib uri="um.tld"`	`<um:getTopLevelGroups>`
	`<um:getusernames>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:getUsernames>`
	`<um:getusernamesforgroup>`	Camel case `taglib uri="um.tld"`	`<um:getUsernamesForGroup>`
	`<um:removegroup>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:removeGroup>`
	`<um:removegroupfromgroup>` *(previously undocumented)*	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:removeGroupFromGroup>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<um:removeuser>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:removeUser>`
	`<um:removeuserfromgroup>` *(previously undocumented)*	Camel case `taglib uri="um.tld"` attribute `resultId = result`	`<um:removeUserFromGroup>`
User / Security	`<um:login>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:login>`
	---	**New**	`<um:logout>`
	`<um:setpassword>`	Camel case `taglib uri="um.tld"` Attribute `resultId = result`	`<um:setPassword>`
WLPS Utilities	`<es:condition>`	Tag no longer supported. Requires manual replacement. See below - **Note 2: <es:condition>**.	
	`<es:counter>`	`taglib uri="es.tld"` Attribute `id` returns an Integer or Long object. You can no longer change the value of the `counter` variable `"id"`. See below - **Note 3: <es:counter>**. Optional attribute `type` can be `long` or `Long` or `Integer` or if not specified is assumed to be `Integer`.	`<es:counter>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<es:date>`	`taglib uri="es.tld"`	`<es:date>`
	`<es:foreachinarray>`	Camel case `taglib uri="es.tld"` Attribute `array` must be a run-time expression (`<%=expression%>`). Attribute `counterId` returns an Integer object (use `id.intValue()`).	`<es:forEachInArray>`
	`<es:isnull>`	Camel case `taglib uri="es.tld"` Attribute `id = item` Attribute `item` must be a run-time expression. An empty string is now treated as a value. (An empty string is not null.)	`<es:isNull>`
	`<es:monitorsession>`	Camel case `taglib uri="es.tld"`	`<es:monitorSession>`
	`<es:notnull>`	Camel case `taglib uri="es.tld"` Attribute `id = item` Attribute `item` must be a run-time expression. An empty string is now treated as a value. (An empty string is not null.)	`<es:notNull>`

Table 5-2 Tag Changes for WebLogic Personalization Server 3.1 (Continued)

Library	Old Style Tag Name	Change	New JSP 1.1 Tag
	`<es:preparedstatement>`	Camel case `taglib uri="es.tld"` Add two new scriptlets. See below - **Note 4: <es:preparedStatement>**. Attribute `pool` no longer supported. See below - **Note 4: <es:preparedStatement>**.	`<es:preparedStatement>`
	`<es:simplereport>`	Camel case `taglib uri="es.tld"` Attribute `resultSet` must be a run-time expression.	`<es:simpleReport>`
	`<es:transposearray>`	Camel case `taglib uri="es.tld"` Attribute `array` must be a run-time expression.	`<es:transposeArray>`
	`<es:usertransaction>`	Tag no longer supported. Replace with new scriptlets. See below - **Note 5: <es:usertransaction>**.	
	`<es:uricontent>`	Camel case `taglib uri="es.tld"`	`<es:uriContent>`

WLS Utilities	`<wl:process>`	`taglib uri="weblogic.tld"`
		`notname = notName`
		`notvalue = notValue`
			`<wl:process>`

	---	**New**	`<wl:repeat>`

Additional Notes About Existing JSP Tags

Note 1: <pz:> Tags

To use the `<pz:div>` and `<pz:contentSelector>` tags, you no longer need to have the JSP extended. You are no longer required to insert the following directive into your code:

```
 <%@ page extends="com.beasys.commerce.axiom.p13n.jsp.P13NJspBase"
%>.
```

(If you have already added this code, it does no harm to leave it.)

Note 2: <es:condition>

The `<es:condition>` tag is no longer supported. Replace it manually with a scriptlet, creating your own `if` statement.

Old Usage

```
<es:condition test="schemaPortletNames.length>0"> </es:condition>
```

New Usage

```
<% if (schemaPortletNames.length>0) { %>
<% } %>
```

Note 3: <es:counter>

If you were manipulating the `counter` variable within the `<es:counter>` tag, you will now need to use a scriptlet instead.

Old Usage

```
<es:counter id="colIter" minCount="0"
maxCount="<%=numOfCols%>">
colIter++;
</es:counter>
```

New Usage

```
<%
for (int colIter = 0; colIter<numOfCols; colIter++) {
colIter++;
}
```

```
%>
```

Note 4: <es:preparedStatement>

The new `<es: preparedStatement>` tag includes two new scriptlets. In addition, this tag no longer supports the `"pool"` attribute. (The pool defined in `commerce.properties` as `"commerce.jdbc.pool.name"` is used for connections.)

Old Usage

```
<es:preparedstatement id="ps" sql"<%=bookmarkBean.QUERY%>"
pool="commercePool">
<%
    bookmarkBean.createQuery(ps, owner);
    java.sql.ResultSet resultSet = ps.executeQuery();
    bookmarkBean.load(resultSet);
%>
</es:preparedstatement>
```

New Usage

```
<es:preparedStatement id="ps" sql="<%=bookmarkBean.QUERY%>">
<%@ include file="startPreparedStatement.inc" %>
<%
    bookmarkBean.createQuery(ps, owner);
    java.sql.ResultSet resultSet = ps.executeQuery();
    bookmarkBean.load(resultSet);
%>
<%@ include file="endPreparedStatement.inc" %>
</es:preparedStatement>
```

Note 5: <es:usertransaction>

The old `<es:usertransaction>` tag is no longer supported. The following code illustrates how to create equivalent functionality.

Old Usage

```
<es:usertransaction>
---- body of page --------
</es:usertransaction>
```

New Usage

```
<%
setSessionValue(com.beasys.commerce.axiom.jsp.JspConstants.
```

```
USER_TRANS_TIMEOUT, "500",request);
// tx timeout defaults to 600 sec. without above line
%>
<%@ include file="startUserTransaction.inc" %>
---- body of page --------
<%@ include file="endUserTransaction.inc" %>
```

New JSP Tags in Release 3.1

This section covers the *new* tags in this release; see "Migrating Existing JSP Tags" on page 5-89 for *changed* tags and migration information.

New Property Set Management Tags in Release 3.1

Two new Property Set Management JSP extension tags provide the following services:

- List all properties associated with a property set.
- List all property set names for a property set group name (for example, `USER` or `CONTENT`).

The two new Property Set tags are:

- <ps:getPropertyNames> – Returns a list of property names for a given property set in a String array.
- <ps:getPropertySetNames> – Returns a list of property set names for a given schema group name in a String array.

New Internationalization Tags in Release 3.1

In earlier releases of WebLogic Personalization Server, Internationalization (I18N) was applied from JSP beans that supported sample portal pages, and administration tools pages. The JSP beans employed a simple MessageBundle Java class that allowed access to localized text labels and messages.

For this release, this basic MessageBundle has been extended using a simple framework that is accessible from JSPs via a small I18N extension tag library. The JSP extension tag library provides the following services:

- Retrieves a static text label or a message from a resource bundle (implemented as a property file).

- Initializes a page context with a particular language, country, and variant for label and message retrieval throughout a page.
- Properly sets the content type (text/html) and character encoding for a page.

The following new tags are included in the I18N framework:

- <i18n:localize> – Allows you to define the language, content type, and character encoding to be used in a page. It also allows you to specify a country, variant, and resource bundle name to use throughout a page when accessing resource bundles via the `<i18n:getMessage>` tag described below.
- <i18n:getMessage> – Retrieves a localized label, or message (based on the absence/presence of an "args" attribute). This tag optionally takes a bundle name, language, country, and variant to aid in locating the appropriate properties file for resource bundle loading.

New WebLogic Utility Tag in Release 3.1

- <wl:repeat> – This WebLogic Server tag is used to iterate over a variety of Java objects that includes:
  - Enumerations
  - Iterators
  - Collections
  - Arrays
  - Vectors
  - Result Sets
  - Result Set Metadata
  - Hashtable keys

Migrating Data From 2.0.1 to 3.1

This section covers upgrading WebLogic Personalization Server database schemas from 2.0.1 to 3.1.1

Note: To upgrade from 2.0.1 to 3.2, see "Migrating Data" on page 5-85 of *Migrating WebLogic Commerce Server From Version 3.1.1 to Version 3.2*.

In WebLogic Personalization Server Release 3.1, a new column called `PROFILE_TYPE` was added to the WLCS_USER table. This column contains the name of the Unified Profile Type of which the User is an instance. (For more information about Unified Profile Types, see the chapter "Creating and Managing Users"in the *Guide to Building Personalized Applications*)

The `PROFILE_TYPE` column can be added to existing WLCS_USER tables with the following statement:

```
ALTER TABLE WLCS_USER ADD PROFILE_TYPE VARCHAR2(100);
```

For User objects that are of the standard type com.beasys.commerce.axiom.contact.User, this should be left as null. If the User is an extended User type, such as the 'Unified Profile Example', the column should be set to that type name. The example user for the Unified Profile Example should be updated with the following statement:

```
UPDATE WLCS_USER SET PROFILE_TYPE = 'Unified Profile Example' WHERE
IDENTIFIER = 'unifieduser_bob';
```

Note: In this document, `$WL_COMMERCE_HOME` refers to the directory into which you installed Commerce services and/or WebLogic Personalization Server and *`database-type`* refers to the type and version of RDBMS that you installed.

To upgrade a WebLogic Personalization Server database from release 2.0.1 to release 3.1, follow these steps:

1. Make a backup copy of the following file:
   `$WL_COMMERCE_HOME/db/`*`database-type`*`/staging/upgrade-to-310.sql`
2. Open `upgrade-to-310.sql` in a text editor.
3. Move the cursor immediately below the following statement:

   ```
   ALTER TABLE WLCS_USER ADD (
        PROFILE_TYPE      VARCHAR2(100)
   );
   ```

4. For each user that is of an extended User type, add the following statement on a single line:

   ```
   UPDATE WLCS_USER SET PROFILE_TYPE = '<profile-type>' WHERE
   IDENTIFIER = '<user-name>';
   ```

For example:

```
UPDATE WLCS_USER SET PROFILE_TYPE = 'Unified Profile Example'
WHERE IDENTIFIER = 'unifieduser_bob';
```

5. Save your modifications and close `update-to-310.sql`.
6. Run the following SQL command:
   `@ $WL_COMMERCE_HOME/db/`*`database-type`*`/update-to-310.sql`

   For example, if you installed Commerce services in `~/WebLogicCommerceServer3.2`, enter the following from SQL*Plus:
   `@ ~/WebLogicCommerceServer3.2/db/`*`database-type`*`/update-to-310.sql`

When the command successfully completes upgrading the database, it prints the following message:

```
************  SCRIPT HAS FINISHED EXECUTING   *************
******  PLEASE REVIEW UPDATE-TO-310.LOG  FILE  *****
```

Verifying the Migration to 3.1

In a test environment, run a typical selection of common user and administrator tasks to be sure your system was migrated correctly. Suggested tasks include the following:

- Go to the example portal and be sure you can log in. See "Overview of the Personalization Tour" in the *Personalization Server Tour* documentation.
- Go to the example Commerce Server application and make sure you can both view items and add them to a shopping cart. See "Registered User Buys a Product" in the *Tour of the JSP Templates* documentation.
- Log into the administration tools and view properties for several users. See "Working with User Profiles" in the *Personalization Server Tour* documentation.

Refer to the appropriate section of the referenced documentation and this guide if you encounter errors.

What's Next: Getting Started With 3.1

Use the online documentation site to locate the documentation you need for additional learning.

http://edocs.bea.com/wlcs/docs31/index.htm

For an overview of how the products work together, see the *Product Family Overview* at http://edocs.bea.com/wlcs/docs31/intro/index.htm.

Index

C

D

E

F

N

O

P

R

S

T

U

V

W

X