



BEA WebLogic Personalization Server

User's Guide

BEA WebLogic Personalization Server 3.1.1
Document Edition 3.1.1
August 2001

©Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

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

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

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

Trademarks or Service Marks

BEA, WebLogic, Tuxedo, and Jolt are registered trademarks of BEA Systems, Inc. How Business Becomes E-Business, Operating System for the Internet, Liquid Data, BEA WebLogic E-Business Platform, BEA Builder, BEA Manager, BEA eLink, BEA WebLogic Commerce Server, BEA WebLogic Personalization Server, BEA WebLogic Process Integrator, BEA WebLogic Collaborate, BEA WebLogic Enterprise, BEA WebLogic Server, BEA WebLogic Integration, E-Business Control Center, BEA Campaign Manager for WebLogic, and Portal FrameWork are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

BEA WebLogic Personalization Server User's Guide

Document Edition	Date	Software Version
3.1.1	August 2001	BEA WebLogic Personalization Server 3.1.1

Contents

About This Document

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

1. Overview of the WebLogic Personalization Server

What Is WebLogic Personalization Server?	1-2
The Advisor	1-3
User Management	1-3
Rules Management	1-4
Property Set Management	1-5
Content Management	1-5
Portal Management	1-6
Foundation Classes and Utilities	1-7

2. Creating and Managing Portals

Introduction to Portals and Portlets	2-2
The Portal Framework	2-3
What is the Difference Between a Portal and a Portlet?	2-3
Personalizing a Portal	2-4
Setting Up	2-5
Logging On to the Administration Tool	2-5
Configuring the Flow Manager to Control Portal Access	2-6
_DEFAULT_PORTAL_INIT property set	2-6
Valid Flow Manager Parmaters	2-7

Creating a Portal Web Site Directory Under the Server Document Root ..	2-9
Using the Portal Administration Tool	2-11
Administering Portlets.....	2-12
Creating Portlets.....	2-14
Editing Portlets	2-16
Deleting Portlets	2-17
Administering Portals.....	2-18
Creating Portals	2-18
Editing Portals.....	2-19
Editing Portal Definitions.....	2-20
Adding and Removing Portlets	2-20
Editing Portlet Display Attributes	2-21
Editing the Portal Layout	2-22
Editing the Portal Color Scheme	2-22
Associating Groups with a Portal	2-23
Deleting Portals	2-24
Administering Portal Groups.....	2-25
Editing Portal Groups	2-25
Adding and Removing Portlets from a Portal Group	2-25
Editing the Portal Group Layout	2-26
Editing the Portal Group Color Scheme	2-27
Creating a Portal Using the Acme Demo	2-29
Introducing the Acme Demo Portal.....	2-29
Starting the Acme Demo Portal.....	2-30
Building the Acme Demo Portal Components	2-31
Creating Portlets for Your Demo Portal.....	2-32
Associating Portlets with Your Demo Portal	2-36
Editing Your Demo Portal Layout	2-37
Editing Your Demo Portal Color Scheme.....	2-38
Testing Your Demo Portal.....	2-39
Where to Get More Information.....	2-40

3. Creating and Managing Property Sets

Overview of Property Sets.....	3-2
Property Value Retrieval via ConfigurableEntity	3-6

Using the Property Set Management Tool	3-8
Creating Property Sets.....	3-8
Creating Properties Within a Property Set.....	3-9
Setting Up the Property Default Value	3-10
Editing Property Sets.....	3-12
Editing Properties Within a Property Set	3-12
Deleting Property Sets.....	3-13
Deleting Properties	3-13

4. Creating and Managing Users

Overview of User Management.....	4-2
Users and Groups	4-3
Unified User Profiles	4-5
Configuration 1	4-6
Configuration 2	4-7
Configuration 3	4-8
Configuration 4	4-9
Using WebLogic Realms.....	4-20
Anonymous User Profiles.....	4-22
Privacy Statement	4-23
User Manager	4-24
Using the User Management Tool.....	4-26
Creating Groups	4-27
Deleting Groups	4-28
Adding Users to Groups.....	4-29
Removing Users from Groups.....	4-31
Editing Group Property Values	4-32
Creating Users	4-33
Editing User Property Values.....	4-34
Deleting Users	4-36
Creating Unified Profile Types	4-37
Editing Unified Profile Types	4-39
Deleting Unified Profile Types	4-39
Using Other Realms	4-40
Registering User Attributes for Retrieval from LDAP	4-40

Unregistering User Attributes for Retrieval from LDAP	4-41
Registering Group Attributes for Retrieval from LDAP	4-41
Unregistering Group Attributes for Retrieval from LDAP	4-42
Viewing LDAP Configuration Settings.....	4-43
Selecting Groups for Use in the Personalization Server from the Realm	4-44
Mapping Realm Groups to the Personalization Server	4-45
Deleting Groups from Your Database.....	4-45
Deleting User Records That Do Not Exist in the Realm from the Personalization Database.....	4-47

5. Creating and Managing Content

What Is the Content Manager?	5-2
Using Third-party Tools	5-4
How Do I Choose What Content Management Tools to Use?	5-4
Constructing Queries Using Java	5-5
Differences Between Content Management and Document Management.	5-5
Using the Document Servlet.....	5-6
Example 1: Usage in a JSP.....	5-7
Example 2: Usage in a JSP.....	5-7
JSP Tags	5-8
Configuring the Content Manager	5-8
Configuring the Document Schema EJB Deployment Descriptor	5-9
Configuring the DocumentManager EJB Deployment Descriptor	5-10
Setting Up Connection Pools.....	5-11
Example connection pool Entry	5-12
Configuring WebLogic Commerce Properties	5-13
Using the Show Document Servlet.....	5-14
Querying Document Content.....	5-14
Structuring a Query	5-15
Using Comparison Operators to Construct Queries	5-17
Using the BulkLoader to Load File-based Content.....	5-18
Command Line Usage.....	5-18
How the BulkLoader Finds Files	5-21
How the BulkLoader Finds Metadata Properties	5-22
Cleaning Up the Database	5-23

Loading Internationalized Documents	5-23
Generating Schema Files	5-24
Using Content Management JSP Tags	5-25
Content Cache	5-25
readOnly Content Tag	5-26
Object Interfaces	5-26

6. Creating and Managing Rules

What Is the Rules Manager?	6-2
Well-known Objects	6-3
How the Rules Engine Works	6-3
What Are Rule Sets?	6-4
Classifier Rules	6-5
Adding “and” conditions to a classifier rule	6-5
Creating Classifier Rules Based on Request Properties	6-6
Content Selector Rules	6-8
Debugging Rule Sets	6-10
What Is the Relationship Between Property Sets and Rules?	6-11
Content Type and Content Selector Rules	6-11
Using the Rules Management Administration Tool	6-12
Creating a Rule Set	6-12
Opening a Rule Set	6-13
Editing Rule Set Properties	6-13
Saving a Rule Set	6-14
Deleting a Rule Set	6-14
Navigating Between Rule Types	6-15
Finding a Rule	6-15
Creating a Classifier Rule	6-16
Editing a Rule	6-17
Editing Rule Properties	6-17
Adding an If User Phrase to a Rule	6-18
Editing an If User Phrase	6-21
Deleting a Rule Phrase	6-21
Creating a Content Selector Rule	6-22
Adding an If User Classifier to a Content Selector Rule Phrase	6-23

Adding an And When Phrase to a Content Selector Rule	6-24
Adding a Then Display Content Phrase to a Content Selector Rule	6-25
Editing a Then Display Content Phrase.....	6-27

About This Document

This document explains how to use the BEA WebLogic Personalization Server™ to create personalized applications for use in an e-commerce site.

This document includes the following topics:

- [Chapter 1, “Overview of the WebLogic Personalization Server,”](#) provides an overview of all the tools and components provided with within the BEA WebLogic Personalization Server.
- [Chapter 2, “Creating and Managing Portals,”](#) describes how to create personalized application content on the Internet.
- [Chapter 3, “Creating and Managing Property Sets,”](#) discusses how Property Set Management allows you to create property sets, the schema of personalization attributes, and the properties that make up property sets.
- [Chapter 4, “Creating and Managing Users,”](#) discusses how User Management joins enterprise data about users with profile data that is used to personalize the user’s view of the application.
- [Chapter 5, “Creating and Managing Content,”](#) discusses how the Content Management component provides content and document management capabilities for use in personalization services. You can use the document management system (DMS) provided as a reference implementation with WebLogic Personalization Server, or you can use a third-party vendor’s DMS, including the Interwoven TeamSite/OpenDeploy product and the Documentum 4i product.
- [Chapter 6, “Creating and Managing Rules,”](#) discusses how the Rules Management component allows developers to create business rules that turn on and off content and match content to users according to their profile information.

What You Need to Know

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

- *Java Server Page (JSP) developer* creates JSPs using the tags provided or by creating custom tags as needed.
- *System analyst or database administrator* writes rules, designs schemas, optimizes SQL and monitors usage.
- *System administrator* installs, configures, deploys, and monitors the Web application server.
- *Java developer* extend or modifies the Enterprise Java Bean (EJB) components that make up the Commerce Server engine, if that level of customization is needed.

e-docs Web Site

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

How to Print the Document

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

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

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

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Personalization Server 3.1 release.

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

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

- Your name, e-mail address, phone number, and fax number
- Your company name and company address

- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.
monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> #include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float
monospace boldface text	Identifies significant words in code. <i>Example:</i> void commit ()
<i>monospace italic text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> LPT1 SIGNON OR
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. <i>Example:</i> buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.



1 Overview of the WebLogic Personalization Server

This chapter is about provides an overview of all the tools and components provided with the BEA WebLogic Personalization Server.

This topic includes the following sections:

- What Is WebLogic Personalization Server?
- The Advisor
- User Management
- Rules Management
- Property Set Management
- Content Management
- Portal Management
- Foundation Classes and Utilities

What Is WebLogic Personalization Server?

The WebLogic Commerce Server is a complete solution for building personalized e-commerce sites. Within the WebLogic Commerce Server (WLCS), personalization functionality is delivered by the WebLogic Personalization Server (WLPS).

Personalization is the means by which Web content developers can tailor an application to a particular individual or group based on any number of criteria. The criteria can be predefined user attributes such as age and gender, or can be based on behavioral information gathered as the user navigates a site.

Using WebLogic Personalization Server, you can build Java-based Internet pages and sites with dynamic, personalized document content. You can customize what content gets delivered based on individual user profiles. WebLogic Personalization Server has a built-in rules editor that you use with JavaServer Pages (JSP) tags to deliver a responsive, customized experience for users.

With WebLogic Personalization Server you can build a wide range of portal types, from business-to-consumer “megaportals” to business-to-business enterprise portals. What this means for your e-commerce enterprise is a flexible, personalized Web presence that listens and responds to your customers and partners based on what you define as important factors.

WebLogic Personalization Server makes extensive use of J2EE mechanisms such as Java Server Pages (JSP) with tag library extensions, session and entity Enterprise Java Beans (EJBs), and Java Naming Directory Interface (JNDI).

WebLogic Personalization Server is a complete solution that enables rapid deployment of adaptable and personalized applications, allowing your businesses to extend competitive advantage and accelerate response time to customer and market demands.

The Advisor

The Advisor ties together the services and components in the WebLogic Personalization Server to deliver personalized content. It does this by matching content to information contained in the user profile. In the rest of this chapter, we will look at each of the WebLogic Personalization Server services in turn, but first let's look briefly at the Advisor.

The Advisor provides an embedded rules engine to classify a user and create a dynamic query into a content database to return *personalized* content for that user. The Advisor is an EJB that can be accessed directly, but is typically accessed through a set of JSP tags. These JSP tags allow HTML developers to assemble dynamic pages without writing any Java code. The Advisor was built to scale for large e-commerce Web sites. It uses a pooling and caching mechanism to keep rules engines ready for rapid evaluation of Personalization rules. For more information about the Advisor, see the chapter [“Creating Personalized Applications with the Advisor”](#) in the WebLogic Personalization Server *Developer's Guide*.

User Management

In the WebLogic Personalization Server, user management is provided through a set of administration tools, or alternatively through any WebLogic Server Realm. Typically, new sites that rely on self-registration will use the WebLogic Personalization Server user management tools. For sites with large collections of existing users in the form of customers or employees, the WebLogic Realm support can provide added security through products such as LDAP servers.

For more information about User Management, see [Chapter 4, “Creating and Managing Users.”](#)

Rules Management

A rule can be thought of as a stylized if-then construct. In the WebLogic Personalization Server, rules are used to do two things: 1) classify users and 2) select content for display from the content management system based on the user. A rules service is provided to take the appropriate inputs for a particular user and return results.

A classification rule may be employed to determine if the user logged in fits a certain classification, given the user's and current group's property values or the `Request` or `Session` properties. So, for example, if a user is over 35, under 65, and is male, then the user is considered a middle age man. If a classifier rule evaluates to true, it returns a `Classification` object with the same name as the classification. The results of this rule can be used by a page developer to vary the content displayed based on one or more classifications.

Content selector rules, if they evaluate to true, will result in a query that can be sent to the content management component. The *if* parts of the content selectors can make decisions based on all the same types of criteria as the classification rules, and also can use the current time to constrain when the rule is in effect. They can also refer to classifier rules, so fundamental categorizations can be reused. The result of a content selector is a `ContentQuery` object, which contains a query expression that can be directly submitted to the content management component.

Rules in WebLogic Personalization Server are organized and saved in rule sets. A rule set may have any combination of classifiers and content selectors, which can be called by name using the JSP tags.

For more information on rules, see [Chapter 6, "Creating and Managing Rules."](#)

Property Set Management

In the Property Set Management tool, you create property sets and define the properties that make up those property sets. Property Set Management provides schema details to personalization server subsystems such as Rules Management and Portal Management. Group, user, request and session schemas can be created in Property Set Management. Such profile schemas prescribe sets of profile attributes. A property can be considered a name/value pair. Property sets serve as namespaces for properties, so that properties can be conveniently grouped, and so that multiple properties with the same name can be defined.

For more information about property sets, see [Chapter 3, “Creating and Managing Property Sets.”](#)

Content Management

The content that is displayed in Web pages may be simple or complex, statically or dynamically determined, and stored in a variety of ways.

- A simple type of content may be a GIF file; a complex type might be a “parse and display” of financial data from an external database query.
- A static piece of content may be an included HTML page with a company logo; a dynamic type may be a set of data representing stock quotes based on user preferences and the current quote value.
- Example sources of content might include: JSP, static HTML, XML, and files in many other formats; queries of external and internal database systems; HTTP and FTP results from other servers; or the content management system built into WebLogic Personalization Server.

WebLogic Personalization Server provides a Content Management component, which is essentially a storage site on the disk. Content that needs to be queried in a dynamic or complex way, and content that changes over time, are good candidates for storage

using this component. Content stored in the content management component may be queried directly from JSP, through specialized query tags, or may be used by other components.

The JSP developer determines the content types and the metadata properties that will be associated with the content type. (Note that content properties are managed by the Content Management system, not the Property Set Management system.) A standard set of metadata properties, such as author, creation date, and MIME type, are automatically associated with all content types.

WebLogic Personalization Server provides a set of object types to construct queries to be submitted to the Content Management component. The page developer has tags available to do this, and other components use this facility as well. A collection of content objects is returned.

For more information on Content Management, see the chapter [Chapter 5, “Creating and Managing Content.”](#)

Portal Management

Portal Management provides an HTML windowing toolkit that allows Web site developers, group administrators and actual end-users to personalize, customize and individualize the layout, look, and content of an e-commerce site. It provides these features through a set of JSP tags, EJBs and administration tools. JSP developers who choose to use the Portal product will develop *portlets* in JSP. These portlets are mini-windows to information, content or application services that are available in the portal. Portlets can be minimized, maximized in their own window, edited, and provided with help. Once you have built the portlets, a portal administrator can pick and choose portlets and make them available in the portal. Then, a portal user controls the layout and visibility of these mini-windows.

For more information about portals and portlets, see the chapter [Chapter 2, “Creating and Managing Portals.”](#)

Foundation Classes and Utilities

The Foundation component provides a set of objects and utilities to support personalization activities. This component provides EJB objects to support HTTP handling, including object definition for request and session objects, as well as object-based utilities specifically designed to support portals.

Foundation includes the Flow Manager, 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 allows navigation information to move 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.

For more information about the Foundation component, see the chapter “[Foundation Classes and Utilities](#)” in the WebLogic Personalization Server *Developer’s Guide* .

1 *Overview of the WebLogic Personalization Server*

2 Creating and Managing Portals

This chapter discusses the WebLogic Personalization Server Administration Tool , and presents instructions for building an example portal, the Acme Demo.

The WebLogic Personalization Server Administration Tool contains a complete set of functions that enable portal administrators to easily create the components of a portal page and personalize the portal's content, layout, and appearance.

WebLogic Personalization Server includes Acme Demo Portal, a complete demo portal that is set up for you and ready to run. This demo portal uses the [Cloudscape Database Management System \(DBMS\)](#) to store the Portal Demo data. Use the Acme Demo Portal to quick start your portal development.

This section includes the following topics:

- Introduction to Portals and Portlets
 - The Portal Framework
 - What is the Difference Between a Portal and a Portlet?
 - Personalizing a Portal
- Setting Up
 - Configuring the Flow Manager to Control Portal Access
 - Creating a Portal Web Site Directory Under the Server Document Root
 - Using the Portal Administration Tool

- Using the Portal Administration Tool
 - Administering Portlets
 - Administering Portals
 - Administering Portal Groups
 - Creating a Portal Using the Acme Demo
- Creating a Portal Using the Acme Demo
 - Starting the Acme Demo Portal
 - Building the Acme Demo Portal Components
 - Creating Portlets for Your Demo Portal
 - Associating Portlets with Your Demo Portal
 - Editing Your Demo Portal Layout
 - Editing Your Demo Portal Color Scheme
 - Testing Your Demo Portal

Introduction to Portals and Portlets

Internet portals are a key part of many eCommerce applications. Portals provide an entry point to the Internet as well as value-added services such as searching and application integration. The WebLogic Personalization Server allows you to quickly assemble both Business-to-Consumer and Business-to-Business portals that require personalized application content on the Internet.

Intelligent portals can act as tour guides to points of interest, tailored for individual user preferences. There are portals that concentrate on collecting and delivering specialized areas of information such as stock trading and finances, emerging technologies, or corporate information. For example, www.boston.com is a specialized news portal and www.schwab.com is a specialized financial portal. Other *megaportals* provide general channels of information such as health, weather, sports, news, E-mail services, chat rooms, news groups, and so on.

Internet portals are an efficient way to exchange large volumes of information with large groups of people. From the users' perspective, most portals are organized as a hierarchical web site where the main page provides links to a set of pages that provide a more detailed view of the data.

Static portals, like many corporate home pages, provide a standard set of information to everyone who visits the site. In contrast, *dynamic personalized portals*, where the information presented may differ based on who is viewing the portal, represent a far more efficient and targeted way to do business. Well-known examples of dynamic portals include www.amazon.com, www.ebay.com, www.excite.com, and my.yahoo.com. With the WebLogic Personalization Server, you can quickly build powerful, dynamic portals like these, as well as static ones.

The Portal Framework

WebLogic Personalization Server provides a collection of prebuilt JSP pages that provide the core functionality for portals. They are located in `%WL_COMMERCE_HOME%/server/public_html/portals/repository`.

When using the framework, your pages will have a common layout. In this layout, a page's real estate will be divided into three main areas: a header, a content area, and a footer. The header resides at the top of the page and typically contains a full-sized logo for the site plus some navigation features. The footer resides at the bottom of the page and typically contains legal notices, copyright information, and a small logo. The middle section, the content area, contains any number of small independent components called portlets. The JSPs included in the Portal Framework manage the layout of these portlets on the page.

What is the Difference Between a Portal and a Portlet?

A *portlet* is a highly focused channel of information served up by a portal. A portal can contain many of these information channels. For example, an online retail portal could provide a variety of interactive merchandise portlets, each presenting a different specialty category such as mystery books, classical music CDs, and baseball memorabilia. Unlike a static portal page, the deployment of portlets via the

Personalization Server gives our online retailer the ability to dynamically respond to customers based on profiles. With this technology, not only can the retailer provide dynamic content, but also the customer can easily select and arrange their e-Commerce portlets.

For example, a returning customer, Samantha, who loves mystery novels and ghost stories could select the “mystery” portlet as central to her standard view of the retailer’s home page. This would be done by means of an edit page made available by the retailer. At the same time, the retailer could determine that Sam’s purchase choices and portlet selections convey a taste for the unexplained. The Personalization Server lets you incorporate sophisticated rules technology to automatically generate *responses* to user profiles. A response *could* be the delivery of specialized information via a portlet. In the case of the mystery hound, our fictitious retailer could offer up recommendations about the latest thrillers and *whodunnits* on video.

Personalizing a Portal

Personalization allows you to customize your portals and portlets to serve a specific audience and purpose. The WebLogic Personalization Server supports three levels of personalization, all of which can be administered with web-based tools. The three levels of personalization are as follow:

- Portal
- Group
- User

Personalization includes web-based forms for adding and removing portal content, editing the content layout, and customizing the portal content color schemes. The user personalization information includes user information and general user preferences.

Setting Up

To properly create and administer a portal using the Portal Administration Tool, you should know how to configure and run the WebLogic Server, and set up database connections.

To create and administer a portal, first you must install and setup the WebLogic Personalization Server software. Then, you must also complete the following tasks:

- [Configuring the Flow Manager to Control Portal Access](#)
- [Creating a Portal Web Site Directory Under the Server Document Root](#)

Logging On to the Administration Tool

Once you have configured the Flow Manager and created a web site directory, you can log on to the Portal Administration Tool.

To log on to the Portal Administration Tool:

1. Start the WebLogic server configured for portal use.
2. Access `http://hostname:port/tools` in your web browser where 'hostname' is the name of the host running your WebLogic Server, 'port' is the port number at which the WebLogic Server is listening for requests, and `tools` is the name of the webapp in the `weblogic.properties` file.

A dialog box appears and prompts you to enter a username and password.

3. Enter the username `administrator` and the password `password`.

Note: We recommend that after installation you immediately create a new password for the administrator.

4. Click Ok to display the Administration Tool Home page.
5. Click the Portal Administration page icon.

Configuring the Flow Manager to Control Portal Access

The Flow Manager controls user access to your portal. The Flow Manager receives all incoming HTTP requests and dispatches each request to the appropriate destination URL. For more information about the Flow Manager, see the topic “Flow Manager” in the [Foundation](#) chapter of the WebLogic Personalization Server *Developer’s Guide*.

_DEFAULT_PORTAL_INIT property set

To configure the Flow Manager, create a portal property set based on a default application init property set.

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 will use 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 Create.
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.

Valid Flow Manager Parmaters

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 Custom Portals](#) in the WebLogic Personalization Server *Developer’s Guide*.
7. When you have finished setting properties, click the Finished button at the bottom of the page.

The table below lists valid parameters for the Flow Manager servlet.

Table 2-1 Valid Flow Manager Servlet Parameters

Parameter Name	Required	Description
allowautologin	No	Determines whether a client with valid cookies can automatically login. The default is false.

2 Creating and Managing Portals

Table 2-1 Valid Flow Manager Servlet Parameters

Parameter Name	Required	Description
<code>defaultdest</code>	Yes	The default destination page JSP if there is not a valid session for the user. (This page is qualified from <code>yourDocumentRoot</code> as defined in the <code>weblogic.properties</code> file.) To display a default portal page for anonymous users, use: <code>defaultdest=/portals/myPortal/portal.jsp</code> or to force anonymous users to the login page instead of the portal page use: <code>defaultdest=/portals/myPortal/_userlogin.jsp</code>
<code>destinationdeterminer</code>	Yes	Used by the Flow Manager to determine JSP page navigation.
<code>destinationhandler</code>	Yes	Used by the Flow Manager to determine JSP page navigation.
<code>groupName</code>	Yes	The default group name for this portal instance. (When new users register, they are added to this group. This parameter allows you to register two Flow Managers that are alike except for the groups that they service.) This value defaults to everyone.
<code>homepage</code>	Yes	The home page JSP returned by the system in auto-login or from the portal home button. (This page is qualified from <code>yourDocumentRoot</code> as defined in the <code>weblogic.properties</code> file.) Example: <code>homepage=/portals/myPortal/portal.jsp</code>
<code>portalname</code>	Yes	The name given to the portal you created in the Portal Administration Tool. Example: Acme Demo Portal
<code>repositorydir</code>	Yes	Location of default files, (gifs, JSP, etc.)
<code>refreshworkingdir</code>	No	Number of seconds, defaults to -1, which means check every time.

Table 2-1 Valid Flow Manager Servlet Parameters

Parameter Name	Required	Description
<code>sessioncomparator</code>	Yes	How to determine if the session is valid.
<code>timeout</code>	No	Timeout for the cookies or session valued in seconds and defaulting to (-1). If set to (-1), the cookies expire upon exiting the browser. If cookies are disabled, the session invalidates upon browser exit. To retain user login information between browser sessions, set the timeout to a large positive number, such as 999999, and set <code>autologin=true</code> .
<code>t11</code>	Yes	Number of seconds between Flow Manager reexamining this property set.
<code>workingdir</code>	Yes	The working directory JSP for the portal implementation that tells the portal framework where to find your portal pages and the WebLogic Personalization Server pages. (This page is qualified from <code>yourDocumentRoot</code> as defined in the <code>weblogic.properties</code> file.) Example: <code>workingdir=/portals/myPortal/</code>

Creating a Portal Web Site Directory Under the Server Document Root

As a final step before using the Portal Administration Tool, create a web site directory for your portal pages under your WebLogic server document root. Then copy all the files from the demo portal directory to your portal web site directory.

To copy files from the demo portal directory to your web site directory:

1. Using the `myPortal` example in the preceding section, copy the files and subdirectories from:

```
yourDocumentRoot/portals/repository
```

to

```
yourDocumentRoot/portals/myPortal
```

2 Creating and Managing Portals

2. Verify that the Flow Manager `workingdir` property value in the `weblogic.properties` file matches the name of the web site directory to which you just copied the demo portal files. See the Acme Demo Portal Directory and Subdirectories table.

For example, if your server document root is `public_html` and your portal working directory is `/portals/myPortal/`, create the following `workingdir`:

```
public_html/portals/myPortal/
```

Table 2-2 Acme Demo Portal Directory and Subdirectories

Directory	Description
<code>/portals/repository</code>	The portal root directory that contains pages such as <code>header.jsp</code> , <code>footer.jsp</code> , and <code>portalcontent.jsp</code> . see Appendix A, WebLogic Personalization Server Framework Files, for an explanation of these and other JSP files provided with the portal framework.
<code>/portals/repository/images</code>	A directory of images that support your portal and WebLogic Personalization Server components.
<code>/portals/repository/portlets</code>	The directory of all portal JSP and HTML pages and the WebLogic Personalization Server sample portlet applications.
<code>/portals/repository/portlets/images</code>	A directory of images that supports your portlets and WebLogic Personalization Server portlets.

Using the Portal Administration Tool

Figure 2-1 Administration Tool Home Page



Now that you have access to the Portal Administration Tool, you can use it to administer portlets, portals, and business-to-business portal groups. Administrative functions available in the tool include:

- Creating, editing, and deleting portals
- Creating, editing, and deleting portlets
- Personalizing a portal's content, layout, and color scheme at the portal and group levels
- Testing your portal

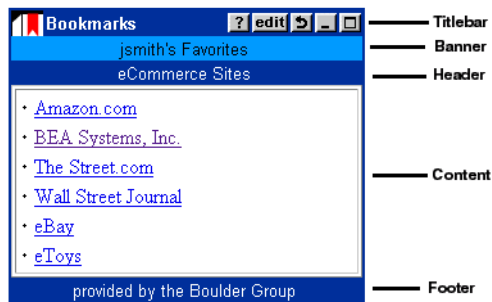
Administering Portlets

To the portal framework, a portlet is a JSP page that knows how to retrieve specialized content and display it in the portal application. To users, a portlet is one of many content modules on a portal page that can be personalized to reflect appearance, content, and layout preferences. Once a portlet is created in the Portal Administration Tool, it can be associated with multiple portals.

You can create, edit, and delete portlets in the Portlets section of the Portal administration tool Home page. All screens in the Administration Tool related to portlet functions are color-coded with teal banners and command buttons. Screens related to portal functions display tan banners and command buttons.

A portlet includes two required components, a titlebar and content area, and several optional components including the banner, header, footer, edit URL, alternate header, alternate footer, maximized URL, and help URL as shown in the following graphic:

Figure 2-2 Portlet Application Decomposed by Components



You can define each portlet application to include any of the following attributes:

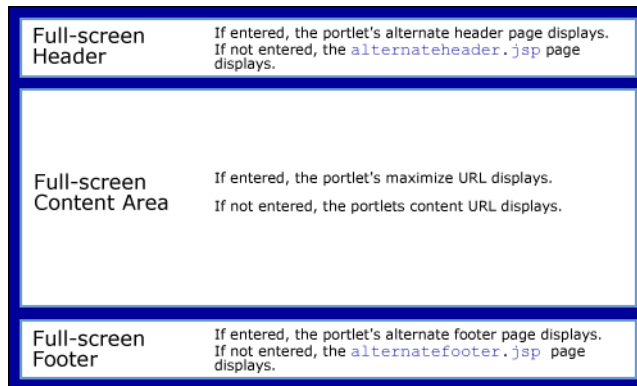
- **Editable**—Enables the user to customize a portlet's content. For example, in a stock portfolio portlet application users can click the Edit icon on the portlet titlebar to access a page that enables them to add or remove stock symbols. If you select this attribute, you must provide an Edit URL.

- **Maximizable**—Allows the portlet to be viewed fullscreen in the browser window. This enables you to provide additional portlet content in the Maximized URL.

The fullscreen page uses:

- **An alternate header**—If no alternate header is specified, the framework uses the default alternate header.
 - **A maximize URL**—If no maximize URL is specified, the framework uses the portlet content area URL as a default.
 - **An alternate footer**—If no alternate footer is specified, the framework uses the default alternate footer.
- When the user clicks the edit or maximize icons in a portlet, the portal framework calls upon its `fullscreen.jsp` page to display in fullscreen mode. The diagram below explains how the `fullscreen.jsp` page determines which content to display.

Figure 2-3 Content Display Criteria



- **Floatable**—Allows the portlet to float on top of the portal screen in a separate browser window. This attribute uses the same header and footer rules as the maximized URL, but displays the content URL instead of the maximized URL.
- **Minimizable**—Reduces the portlet display to the titlebar to minimize the amount of space the portlet occupies on the portal page.

- **Helpable**—Provides a Help icon in the portlet title bar that users can click to access a URL that assists them with the portlet application. If you select this attribute, you must provide a Help URL.
- **Login Required**—Requires the user to be logged on to the portal to view the portlet. For example, if your portal contains a portlet that displays a user's favorite bookmarks, the user must be logged on before the Bookmark's portlet is visible on the portal screen. This attribute helps maintain a secure portal and allows users to retrieve personalized information.
- **Mandatory** —A portlet can now be personalized to be mandatory. A mandatory portlet is one that is always available and visible. The portlet can be made mandatory at the definition, portal personalization, and group personalization levels.
- **Titlebar URL** — A URL can display as the portlet titlebar. It can be a JSP or HTML fragment.

Creating Portlets

Before you use the Portal Administration Tool to create a portlet, place all your portlet application files in the following directory:

```
yourDocumentRoot/portals/repository/portlets
```

You create a portlet in the Administration Tool by creating a portlet definition entity (referred to in this document as a portlet) and associating portlet JSP URLs that have been created by a portlet developer with the portlet entity. When a portlet is created, it is not automatically associated with a portal. You need to add portlets to a portal later from the portal view-page.

To create a portlet:

1. On the Portal Administration Tool Home page, click Create in the Portlets banner. The Create a New Portlet tool displays.
2. Enter the appropriate information in the following required fields:
 - *Portlet Name*—any combination of numbers and letters will be accepted in this field.
 - *Content URL*—enter a URL relative to your portal `workingdir`.

3. If desired, enter the appropriate information in the following optional fields:

- *Header URL*—enter a URL to display as the portlet header. It can be a JSP or HTML fragment.
- *Footer URL*—enter a URL to display as the portlet footer. It can be a JSP or HTML fragment.
- *Titlebar URL*—enter a URL to display as the portlet titlebar. It can be a JSP or HTML fragment.
- *Banner URL*—enter a URL to display as the portlet banner under the portlet titlebar. It can be a JSP or HTML fragment. The following shows a sample banner JSP page:

```
<%@ page extends="com.beasys.portal.admin.PortalJspBase"%>
<%@ page
import="com.beasys.portal.tags.PortalTagConstants"%>
<center>
<font size=-1>To Do's for
<%@
(String)getSessionValue(PortalTagConstants.PORTAL_GROUP,requ
est)%></font>
</center>
```

- *Mandatory*—a portlet can now be personalized to be mandatory. A mandatory portlet is one that is always available and visible. The portlet can be made mandatory at the definition, portal personalization, and group personalization levels.
- *Alternate Header URL*—enter a URL to display as a web page header when the portlet is floated or maximized. If no alternate header exists, the portal framework uses a default called `alternateheader.jsp`.
- *Alternate Footer URL*—enter a URL to display as a web page footer when the portlet is floated or maximized. If no alternate footer exists, the portal framework uses a default called `alternatefooter.jsp`.
- *Editable*—select the check box to enable users to edit a portlet's content. An Edit icon displays in the portlet titlebar. The attribute default is deselected.
- *Edit URL*—if you selected the Editable check box, enter a URL that enables the user to edit the portlet content.

- *Maximizable*—select the check box to enable users to maximize the portlet in the current browser window. A Maximize icon displays in the portlet titlebar. The attribute default is deselected.
 - *Maximized URL*—if you selected the Maximizable check box, enter a URL for the content area of the maximized page. The default URL is your portlet content area URL.
 - *Helpable*—select the check box to enable users to access a help screen. A Help icon displays in the portlet titlebar. The attribute default is deselected.
 - *Help URL*—if you selected the Helpable check box, enter a URL that opens a help topic related to the portlet.
 - *Icon URL*—enter a URL to display an icon (GIF) on the left side of the portlet titlebar. This image should be 27 pixels wide by 20 pixels high with 2 pixels of transparency on the right.
 - *Minimizable*—select the check box to enable users to minimize the portlet in the portal screen. A Minimize icon displays in the portlet titlebar. The attribute default is deselected.
 - *Floatable*—select the check box to enable users to float the portlet in a new browser window. A Float icon displays in the portlet titlebar. The attribute default is deselected.
 - *Login Required*—select the check box to require a user to be logged on to the portal to view the portlet. The attribute default is deselected.
4. Click Create.

If the portlet was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
 5. Click Back to return to the home page. The new portlet name displays under the Portlets banner.

Editing Portlets

After creating a portlet, you can redefine it at any time by adding or removing attributes.

To edit a portlet:

1. On the home page, click a portlet title link to display the Edit Properties tool. The name of the portlet you selected to edit displays at the top of the screen.
2. Enter the appropriate changes.
3. Click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Click Back to return to the home page.

Deleting Portlets

You can delete portlets that you no longer need. However, you must first remove (mark as unavailable) the portlet from any portals it is associated with.

To delete a portlet:

1. On the home page, click Delete in the Portlets banner. The Delete a Portlet tool displays.
2. Select the portlet from the Portlet Name drop-down list.
3. Click Delete. A confirmation window displays.
4. Click Ok to confirm your deletion.
5. Click Back to return to the home page. The portlet name is no longer listed under the Portlets banner.

Administering Portals

You can create, edit, or delete portals from the Portals section of the Portal Administration Tool Home page. All screens in the Administration Tool related to portal functions are color-coded with tan banners and command buttons. Screens related to portlet functions display teal banners and command buttons.

For procedures on using the Acme Demo Portal components to quick start your portal development, see [“Creating a Portal Using the Acme Demo” on page 2-29](#).

Creating Portals

To create a new portal:

1. On the Portal Administration Tool Home page, click Create in the Portals banner to display the Create a New Portal tool.
2. Complete the following required fields:
 - *Portal Name*—any combination of numbers and letters will be accepted in this field.
 - *Content URL*—enter a portal content JSP relative to `workingdir`.
 - *Number of Content Columns*—Enter 1, 2 or 3.
3. Customize your portal display by entering the optional URL files. Make all URLs relative to `workingdir`:
 - *Header URL*—enter a header JSP for the default header page.
 - *Footer URL*—enter a footer JSP for the default footer page.
 - *Suspended*—select the check box to suspend the portal application and replace the portal home page with an 'under maintenance' screen until service resumes. To resume service, deselect the Suspended check box on the Edit Portal Definition tool.
 - *Suspended URL*—enter the default `suspended.jsp` to display the 'under maintenance' URL to end-users while the application is in Suspended mode.
4. Click Create to create the portal definition.

If the portal was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

5. Click Back to return to the home page. The new portal name displays under the Portals banner.

Editing Portals

After creating a portal, you can edit it to associate portlets, groups, and users. You can also personalize the portal's layout and color scheme, and make changes to the definition.

To edit a portal:

1. On the Portal Administration Tool Home page, click a portal title link to see the portal view-page. The name of the portal you selected displays at the top of the screen. Colored banners separate each portal property and contain a command button for that property. See the following procedures for more information on editing portal properties.

The following image shows the portal view-page.

Figure 2-4 Portal View Page

The screenshot shows the Portal Administration Tool interface. At the top, there is a red banner with the 'bea' logo on the left, the text 'Portal Administration Tool' in the center, and three circular icons labeled 'home', a folder icon, and a question mark on the right. Below the banner, a green bar displays 'Portal: Demo Portal' on the left and a 'finished' status indicator on the right. The main content area is divided into sections:

- Definition**: A section with an 'edit' button. It contains the following properties:

Header URL	header.jsp	Columns	3
Content URL	portalcontent.jsp	Suspended	false
Footer URL	footer.jsp	Suspended URL	N/A
- Associated Portlets**: A section with a '+/-' button. It includes a legend 'X=visible portlets' and a table of portlets:

X	Bookmarks	X	Defined Portals	X	Defined Portlets
X	Dictionary	X	Group To Do List	X	My To Do List
X	Quote	X	Search		

2. When you are done viewing and editing the portal, click **Finished** at the top or bottom of the portal view-page to return to the home page.

Editing Portal Definitions

You can edit the portal definition you created to reflect any changes to the associated URLs or number of portal columns.

To edit a portal definition:

1. On the portal view-page, click **Edit** in the **Definition** banner to display the **Edit Portal Definition** tool.
2. Enter the appropriate changes.
3. Click **Save**.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Click **Back** to return to the portal view-page.

Adding and Removing Portlets

You can choose which portlets are available to a portal by adding and removing them from the system's list of all established portlets. From the narrowed list of portlets you associate with a portal, group, and end-users further define which portlets they want available and visible on their personalized portal page.

To associate portlets with a portal:

1. On the portal view-page, click **+/-** in the **Associated Portlets** banner to display the **Add or Remove Portlets** tool.
2. To add a portlet to the portal, select **Avail**. The portlet is associated with the portal. It doesn't appear on the portal page until it is made visible by you, the Group Administrator or the end-user.
3. To make a portlet visible, select **Visible**. The portlet is associated with the portal and now appears on the portal page.

4. To remove a portlet from the portal, select Unavail. The portlet becomes disassociated with the portal and unavailable to new groups and end-users (including anonymous users). However, if the portlet has been personalized at a group or user level, it remains associated with those levels.
5. Click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
6. Click Back to return to the portal view-page. Available portlets appear in the Associated Portlets section of the screen with a gray background. Visible portlets are marked with an check mark.

Editing Portlet Display Attributes

Portlet titles, associated with a portal, display as hot links in the Associated Portlets section of the portal view-page. These links open a tool that enables you to further specify how the portlet displays in the portal, overriding the display attributes established when the portlet was created. Group Administrators can further personalize these attributes.

To edit an associated portlet's display attributes:

1. Click the portlet title link in the Associated Portlets section of the portal view-page.
2. Enter the appropriate changes in the Edit Portlet Display Attributes tool.
3. Click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
4. Click Back to return to the portal view-page.

Editing the Portal Layout

You can move a portal's associated portlets left and right between columns and up and down within columns depending on the column layout you selected when you created the portal. You can also change the percentage of the portal page that each column occupies. Group Administrators and end-users can further personalize the portal layout.

To edit the layout of portlets in a portal:

1. On the portal view-page, click **Edit** in the **Layout** banner to display the **Edit Portal Layout** tool. This layout tool shows each portal column, its span percentage, and the portlets that display within those columns.
2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.
3. Click an arrow to move the portlet up or down within a column, or right or left between columns.

To change the column spans of a portal layout:

1. Click in the percentage field associated with a column and enter a new percentage. The sum of all column spans should equal 100%. For single column portals, you may specify from 1% to 100%.
2. When you are done editing the portal layout, click **Save**.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

3. Click **Back** to return to the portal view-page. A table in the **Portal Layout** section lists the portlets as you arranged them within each column.

Editing the Portal Color Scheme

You can edit the overall appearance of a portal by changing its background color as well as the portlets' component colors, title colors, and border appearance.

To edit portal colors:

1. On the portal view-page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that displays in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the Click here to save changes and preview colors link. Your color changes are saved and the Edit Color Schemes tool redisplay the example portlet at the bottom of the screen to reflect your color preferences.
7. To save your color preferences without previewing them, click Save. The portal view-page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to its original color scheme, click Restore Defaults. The portal view-page displays the default colors associated with the portal in the Colors section of the screen.

Associating Groups with a Portal

You can only associate portal groups from the portal view page. For more information on associating users with a group and personalizing portal groups, see [“Administering Portal Groups” on page 2-25](#).

To associate a group with a portal:

1. On the portal view-page, click +/- in the Associated Groups banner.
2. Expand the hierarchy (or search) to find the desired group.
3. Check the group.
4. Click Save to return to the portal view-page. The new group name displays in the Associated Groups section of the portal view-page.

To disassociated with a group:

1. On the portal view-page, click +/- in the Associated Groups banner.
2. Expand the hierarchy (or search) to find the desired group.
3. Uncheck the group.
4. Click Save to return to the portal view-page. The group name no longer displays in the Associated Groups section of the portal view page.

Deleting Portals

You can delete an existing portal from the Portal administration tool Home page. However, you must first disassociate all portal groups and users from that portal.

To delete a portal:

1. On the home page, click Delete in the Portals banner to display the Delete a Portal tool.
2. Select the portal from the Portal Name drop-down list.
3. Click Delete. A confirmation window displays.
4. Click OK to confirm the deletion.
5. Click Back to return to the home page. The portal name no longer displays in the Portals section of the Portal administration tool Home page.

Administering Portal Groups

The Portal Administration Tool enables you to personalize portal groups. You can personalize the layout, content, and color scheme.

Avoid creating portal groups for business-to-consumer portals with an unmanageable number of users.

You can edit portal groups from the portal view-page.

Editing Portal Groups

Editing portal groups allows you to associate portlets with each group. You can also personalize the group's portal layout and color scheme.

To edit a portal group:

1. On the Portal Administration Tool Home page, click the portal title link the group is associated with. The portal view-page displays.
2. In the Associated Groups section of the screen, click the group title link you want to edit. The group-view page displays. The names of the portal and group you selected display at the top of the screen. Colored banners separate each group property and contain a command button for that property. See the following procedures for more information on editing group properties.

Adding and Removing Portlets from a Portal Group

As the Group Administrator, you choose which portlets are available to a group by adding and removing them from the portal's list of all associated portlets. From the narrowed list of portlets you associate with a group, end-users further define which portlets they want available and visible on their personalized portal page.

To associate portlets with a group:

1. On the group-view page, click +/- in the Associated Portlets banner to display the Add or Remove Portlets tool.

2. To add a portlet to the group, select **Avail**. The portlet is associated with the group. It does not display on the portal page until it is made visible by you or the end-user.
3. To make a portlet visible, select **Visible**. The portlet is associated with the group and displays on the portal page.
4. To remove a portlet from the group, select **Unavail**. The portlet becomes disassociated from the group and unavailable to the group and end-users. However, if the portlet has been personalized at the user level, it remains associated with those users.
5. Click **Save**.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

6. Click **Back** to return to the group-view page. Available portlets appear in the **Associated Portlets** section of the screen with a gray background. Visible portlets are marked with an **X**.

Editing the Portal Group Layout

You can move a group's associated portlets left and right between columns and up and down within columns. End-users can further personalize the portal layout.

To edit the layout of portlets in a group:

1. On the group-view page, click **Edit** in the **Layout** banner to display the **Edit Portal Layout** tool. This layout tool shows each portal column and the portlets that display within those columns.
2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.
3. Click an arrow to move the portlet up or down within a column, or right or left between columns.
4. When you are done editing the portal layout, click **Save**.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

5. Click Back to return to the portal view-page. A table in the Portal Layout section lists the portlets as you arranged them within each column.

Editing the Portal Group Color Scheme

You can edit the overall appearance of a group by changing its portal background color as well as the portlets' component colors, title colors, and border appearance.

To edit group colors:

1. On the group-view page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the Click here to save changes and preview colors link. Your color changes are saved and the Edit Color Schemes tool re-displays the example portlet at the bottom of the screen to reflect your color preferences.

2 *Creating and Managing Portals*

7. To save your color preferences without previewing them, click Save. The group-view page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to its original color scheme, click Restore Defaults. The group-view page displays the default colors associated with the portal in the Colors section of the screen.

Creating a Portal Using the Acme Demo

This section shows you how to use the Acme Demo Portal to quick start your portal development.

This section includes the following topics:

- Introducing the Acme Demo Portal
- Starting the Acme Demo Portal
- Building the Acme Demo Portal Components
- Creating Portlets for Your Demo Portal
- Associating Portlets with Your Demo Portal
- Editing Your Demo Portal Layout
- Editing Your Demo Portal Color Scheme
- Testing Your Demo Portal

Introducing the Acme Demo Portal

When you install the WebLogic Personalization Server, a complete demo portal is set up for you and ready to run. This ready-made demo portal uses the Cloudscape Database Management System (DBMS) to store the Acme Demo Portal data. The demo is ready to run with the Cloudscape database immediately after you complete the WebLogic Commerce Server installation.

Note: The Cloudscape database is included with WebLogic Server under a limited evaluation license. Because of the limitations of the Cloudscape database, other databases should be considered. The database stores all of the portal framework information needed to support the portal components.

Note: The file name for the Acme Demo Portal is `exampleportal`. Throughout the WebLogic Personalization Server documentation, the Acme Demo Portal is referred to by its filename, `exampleportal`, or simply the “example portal.”

The Acme Demo Portal includes the following:

- Portal page JSP templates - header, footer, and portal content layout JSP pages
- Sample portlet applications including portlet JSP pages
- A complete set of end-user portal personalization tools, including:
 - User login and new user registration web forms
 - Change-password and forgot-password web forms
 - End-user personalization tools for customizing portal content
 - Help pages

All of JSP pages for the Acme Demo Portal are located in the following installed product directory: `public_html/portals/repository`.

Starting the Acme Demo Portal

To start the Portal Demo:

1. Start the WebLogic server by executing the `StartCommerce` command file in your installation directory.
2. Open a web browser window.
3. In your web browser ,enter the following demo portal page URL,
`http://hostname:port/application/exampleportal` where 'hostname' is the name of the host running your WebLogic Server, `port` is the port number at which the WebLogic Server is listening for requests, `application` is the name of the Flow Manager servlet in the `weblogic.properties` file, and `exampleportal` is the name of the property set.

Example: `http://mybigbox:7501/application/exampleportal`

You can now use the WebLogic Personalization Server Administration Tool to view the Demo Portal or assemble your own portal, as described in “Using the Portal Administration Tool” on page 2-11.

Notes: Before using the Portal Administration Tool to assemble the Acme Demo Portal, you must install and set up the WebLogic Personalization Server software. For more information, see the *BEA WebLogic Commerce Server with WebLogic Personalization Server [Installation Guide](#)*.

You must also set the WebLogic server document root before you are able to log on to and use the tool. For more information, see “[Creating a Portal Web Site Directory Under the Server Document Root](#)” on page 2-9.

For more information on accessing and using the Administration Tool, see “[Using the Portal Administration Tool](#)” on page 2-11.

Building the Acme Demo Portal Components

To create the Acme Demo Portal definition:

1. Click Create in the Portals banner of the Portal administration tool Home page to display the Create a New Portal tool.
2. Enter the following information in the appropriate fields:

Table 2-3 Sample Data

Field Name	Data
Portal Name	Acme Demo Portal
Header URL	header.jsp
Content URL	portalcontent.jsp
Footer URL	footer.jsp
Number of columns	3

Table 2-3 Sample Data (Continued)

Field Name	Data
Suspend	Defaults to false. Set to true to suspend the portal during maintenance.
Suspended URL	Enter <code>suspended.jsp</code> if you want to display the under maintenance URL during maintenance.

3. Click Create.

If the portal was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Click Back to return to the home page. The name of the new portal, “Acme Demo Portal,” displays in the Portals section of the home page.

Creating Portlets for Your Demo Portal

You create portlets in the Administration Tool by associating a URL with each portlet component. When a portlet is created, it is not automatically associated with a portal. You must add the portlets to the demo portal later from the portal view-page.

The Acme Demo Portal includes six portlet applications that you can assemble with the Portal Administration Tool.

To create the demo portlets:

1. Click Create in the Portlets banner of the Portal administration tool Home page to display the Create a New Portlet tool.
2. To create the first of the demo portlets, My Bookmarks, enter the following information in the appropriate fields:

Table 2-4

Portlet Name	Data
Bookmarks	Portlet Name: Bookmarks
	Content URL: portlets/bookmarks.jsp
	Editable: select the check box
	Edit URL: portlets/bookmarks_edit.jsp
	Maximizable: select the check box
	Icon URL: portlets/images/pt_bookmark.gif
	Login Required: select the check box
	Titlebar URL: Enter a URL to display as the portlet titlebar. It can be a JSP or HTML fragment.
	Mandatory: A mandatory portlet is one that is always available and visible.

3. Click Create.

If the portlet was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Follow steps two through three above to create the remaining five portlets. For each portlet, enter the following information in the appropriate fields:

Table 2-5

Portlet Name	Data
My Dictionary	Portlet Name: My Dictionary
	Content URL: portlets/dictionary.jsp
	Icon URL: portlets/images/pt_dictionary.gif
	Minimizable: select the check box
My To Do List	Portlet Name: My To Do List
	Content URL: portlets/mytodo.jsp
	Editable: select the check box
	Edit URL: portlets/mytodo_edit.jsp
	Maximizable: select the check box
	Icon URL: portlets/images/pt_my_list.gif
	Minimizable: select the check box
Floatable: select the check box	
My Group To Do List	Login Required: select the check box
	Portlet Name: My Group To Do List
	Content URL: portlets/grouptodo.jsp
	Banner URL: portlets/grouptodobanner.jsp
	Editable: select the check box
	Edit URL: portlets/grouptodo_edit.jsp
	Maximizable: select the check box
Icon URL: portlets/images/pt_group_list.gif	
	Minimizable: select the check box

Table 2-5

Portlet Name	Data
	Floatable: select the check box
	Login Required: select the check box
Stock Quote	Portlet Name: Stock Quote
	Content URL: portlets/quote.jsp
	Icon URL: portlets/images/pt_quote.gif
	Minimizable: select the check box
Search	Portlet Name: Search
	Content URL: portlets/search.jsp
	Icon URL: portlets/images/pt_search.gif
	Minimizable: select the check box
News Index	Portlet Name: News Index
	Content URL: portlets/new_index.jsp
News Reader	Portlet Name: News Reader
	Content URL: portlets/news_viewer.jsp
	Titlebar: content_titlebar.jsp

5. Click Back to return to the Home page. The six new portlet names appear in the Portlets section.

Associating Portlets with Your Demo Portal

After creating a portal, you can associate portlets to it. You can also personalize the portal's layout and color scheme, and make changes to the definition. For more information on editing a portal, see [“Editing Portals” on page 2-19](#).

You choose which portlets are available to a portal by adding and removing them from the system's list of established portlets. From the narrowed list of portlets you associate with a portal, group and end-users further define which portlets they want available and visible on their personalized portal page.

To associate the six portlets you just created with the demo portal:

1. On the Portal administration tool Home page, click the Demo Portal title link in the Portals section of the screen. The Acme Demo Portal view-page displays.
2. On the portal view-page, click +/- in the Associated Portlets banner to display the Add or Remove Portlets tool.
3. To add a portlet to the portal, select Avail. The portlet is associated with the portal. It does not display on the portal page until it is made visible by you, the Group Administrator, or the end-user.
4. To make a portlet visible, select Visible. The portlet is associated with the portal and now displays on the portal page.
5. To remove a portlet from the portal, select Unavail. The portlet becomes disassociated with the portal and unavailable to new groups and end-users (including anonymous users). However, if the portlet has already been personalized at a group or user level, it remains associated with those levels.
6. Click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

7. Click Back to return to the Demo Portal view-page. Available portlets appear in the Associated Portlets section of the screen with a gray background. Visible portlets are marked with an X.

Editing Your Demo Portal Layout

You can move a portal's associated portlets left and right between columns and up and down within columns depending on the column layout you selected when you created the portal. You can also change the percentage of the portal page that each column occupies in all portals except group portals. Group Administrators and end-users can further personalize the portal layout.

To edit the layout of portlets in the demo portal:

1. On the Demo Portal view-page, click Edit in the Layout banner to display the Edit Portal Layout tool. This layout tool shows each portal column, its span percentage, and the portlets that display within those columns.
2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.
3. Click an arrow to move the portlet up or down within a column, or right or left between columns.

To change the column spans of a portal layout:

1. Click in the percentage field associated with a column and enter a new percentage. The sum of all column spans should equal 100%.
2. When you finish editing the portal layout, click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

3. Click Back to return to the Demo Portal view-page. A table in the Portal Layout section lists the portlets as you arranged them within each column.

Editing Your Demo Portal Color Scheme

You can edit the overall appearance of a portal by changing its background color as well as the portlet's component colors, title colors, and border appearance.

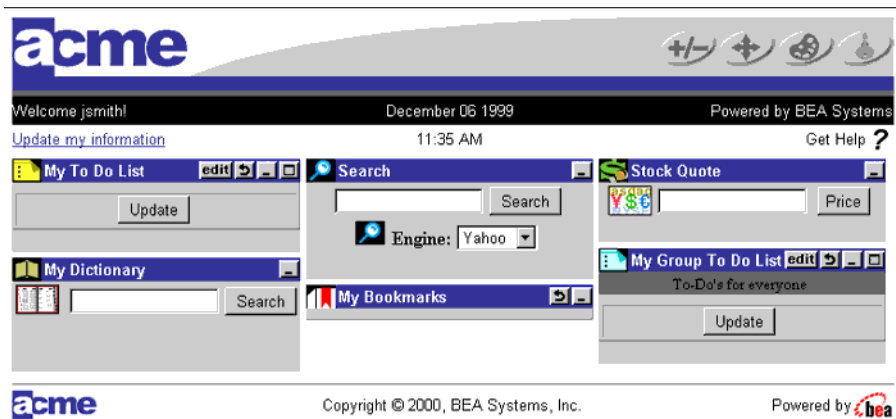
To edit the demo portal colors:

1. On the Demo Portal view-page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that will display in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the "Click here to save changes and preview colors" link. Your color changes will be saved and the Edit Color Schemes tool will re-display the example portlet at the bottom of the screen to reflect your color preferences.
7. To save your color preferences without previewing them, click Save. The portal view-page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to the BEA original color scheme, click Restore Defaults. The portal view-page displays the default colors associated with the portal in the Colors section of the screen.

Testing Your Demo Portal

Once your portal is operational, you should test it to verify that all the associated portlets are available and visible as you specified them, and that your portal displays the correct color scheme and layout.

Figure 2-5 Acme Demo Portal



To test the demo portal:

1. In a web browser, enter the Flow Manager URL:
`http://host:port/application/exampleportal`
The default portal home page should display all visible portlets and should reflect your default color and layout preferences.
2. To test end-user personalization options, sign on to your portal by clicking the Sign On icon in the upper right corner of the home page.

If you have not created a user profile, you can do so by following the registration wizard. If you have created a profile, enter your username and password, and click Sign On.

You can now use the personalization tools to customize the portal's color, layout, and visible portlets.

Figure 2-5 shows the Acme Demo Portal as it displays with the BEA default color scheme and layout to a registered user named jsmith.

Where to Get More Information

This chapter presents the basics of portal development using the WebLogic Personalization Server administration tools. For an in-depth look at portals and portlets, see the following sources:

The chapter “[Building a Custom Portal Step-by-Step](#),” in the WebLogic Personalization Server *Developer's Guide*, presents a tutorial for customizing your own e-Commerce portal.

The chapter “[Developing Portlets](#),” in the WebLogic Personalization Server *Developer's Guide*, presents the application developer with essential information about creating custom portlets.

You may need to consult the following documentation when using the WebLogic Personalization Server:

- <http://edocs.beas.com/wlcs/index.htm>
- [Using WebLogic JSP](#)
- [Using the Cloudscape database with WebLogic](#)
- [Using WebLogic JDBC](#)
- [JSP documentation from JavaSoft at http://java.sun.com/products/jsp](http://java.sun.com/products/jsp)

3 Creating and Managing Property Sets

Property sets are the schemas for personalization attributes. Using the Property Set Management tool, you can create property sets and define the properties that make up these property sets.

This chapter includes the following topics:

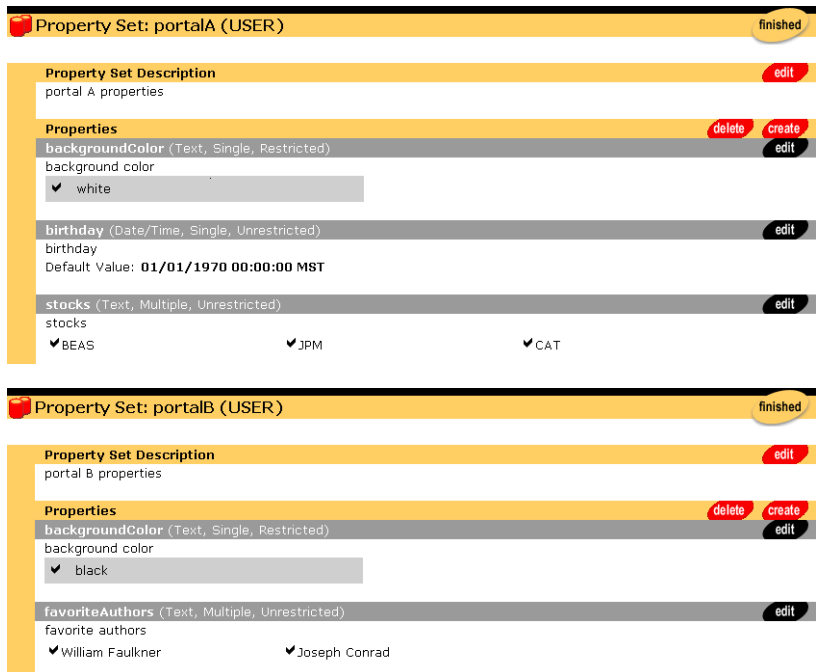
- Overview of Property Sets
- Property Value Retrieval via ConfigurableEntity
- Using the Property Set Management Tool
 - Creating Property Sets
 - Creating Properties Within a Property Set
 - Editing Property Sets
 - Editing Properties Within a Property Set
 - Deleting Property Sets
 - Deleting Properties

Overview of Property Sets

In the most general sense, a property can be considered a name/value pair. Property sets serve as namespaces for properties so that properties can be conveniently grouped and so that multiple properties with the same name can be defined.

For instance, web site developers might want users to be able to specify different background colors for each of their portals by requiring the property “backgroundColor” for a user. By creating “portalA” and “portalB” property sets, the property “backgroundColor” can exist for both portalA and portalB. While the two “backgroundColor” properties have the same name, they could have the same or different definitions. Figure 3-1 shows two property sets with redundant property names, corresponding to unique definitions.

Figure 3-1 Property Sets Serving as Namespaces.



A property definition includes the following information:

- **Property Value Type:** The data type of the property value, for example, Text, Integer, Float, or Date/Time. A property called `age` might be an Integer type, while `lastName` would be Text.
- **Plurality:** Whether the property can contain a single value, or multiple values. A property called `firstName` might be a single-valued property, while `childrenNames` would most likely be multivalued.
- **Restriction:** Whether the allowable values for a property are restricted. A property called `favoriteDayOfTheWeek` would only have seven possible values, while `email` would most likely be unrestricted.
- **Default Property Value:** Default values provided by the property set corresponding to the property. A property called `favoriteDayOfTheWeek` might have a default value of “Saturday.” A property called `daysOff` might have the defaults “Saturday” and “Sunday.”

For Personalization Server purposes, property sets are applied to six major areas.

1. User and Group Profiles

The User/Group property set type is used for defining the property sets and properties that apply to user and group profiles. For example, a property set of this type might be created called `portalA`. Subsequent property retrieval for a particular user or group can then be scoped with this property set name to retrieve the user’s background color for the portal. See [Chapter 4, “Creating and Managing Users,”](#) for an in-depth discussion of how property retrieval works for users and groups.

2. HTTP Sessions

The Session property set type is used for defining the property sets and properties that apply to HTTP sessions. Like the User/Group property set type, a “Session” property set type might be called “`portalA`”. Properties available through this property set can then be accessed via the Advisor.

3. HTTP Requests

The Request property set type is used for defining the property sets and properties that apply to HTTP requests. Again, like the “User/Group” property set type, a “Request” property set type might be called “`portalA`.” Properties available through this property set can then be accessed via the Advisor.

4. Content Management

The Content Management property set type is used for defining the configuration and run-time use of the Content Management system. Content Management property sets cannot be created or manipulated with the Personalization Server administration tools. See [Chapter 5, “Creating and Managing Content,”](#) for more complete information on this subject.

5. Application Initialization

The Application Init property set type uses default values to define application initialization parameters. These are the property sets used by the Flow Manager in support of portal (DEFAULT_PORTAL_INIT) and non-portal (DEFAULT_APP_INIT) based personalized applications. For more information about the Flow Manager, see the chapter [Foundations](#) in the WebLogic Personalization Server *Developer’s Guide*

6. Catalog Custom Attributes

You can define a property set that establishes custom attributes for a product item in the WebLogic Commerce Server catalog. For a given product item, a custom attribute that you define can be used in addition to the default attributes provided by WebLogic Commerce Server in the catalog database tables. For more information, see [“Catalog Administration Tasks”](#) in *Product Catalog Management*.

Creating a property set is a simple task via the Property Set Management tools. A name for the set must be provided as well as description. Properties can be copied from an existing property set if a pre-existing property set defines similar properties. Expanding the previous example, if portalA’s properties have been defined and portalB is going to have the same (or similar) properties, then you can copy the properties from portalA’s property set when creating portalB’s property set. Finally, the type of property set (“User/Group”, “Session”, or “Request”) must be chosen.

When defining a property, specify the following:

- Property name—the name of the property, such as `backgroundColor`.
- Description—a textual description of the property, perhaps describing the purpose of the property.
- Type—the data type of the property value. Data types supported by the administration tools are Text, Integer (equivalent to Long in Java),

Floating-Point number (equivalent to Double in Java), Boolean, and Date/Time (equivalent to java.sql.Timestamp).

- Selection option—determines whether the property is single-valued or multi-valued.
- Creation category—determines whether the possible values are restricted. Restricted property values are restricted to values listed in the property definition. Unrestricted property values have no such limitation.

The following table lists the property definition attribute and value.

Property Definition Attribute	Attribute Value
Name	Text (100 character length maximum)
Description	Text (255 character length maximum)
Type	Text, Integer (equivalent to Long in Java), Floating-Point Number (equivalent to Double in Java), Boolean, or Date/Time
Selection Option	Single-valued or multi-valued
Creation Category	Restricted or unrestricted
Default Value	Up to the user—can be null

Once created, User/Group property values can be edited for a particular user or group via the User Management user and group tools. For “Session” and “Request” properties, the only editable values are the default values set in the property definitions—runtime values are determined by values in the HTTP session or HTTP request, respectively.

Property Value Retrieval via ConfigurableEntity

Property Sets created with the administration tools are stored as `com.beasys.commerce.foundation.property.Schema` components. The component that acts as an “owner” of properties associated with Property Sets is the `com.beasys.commerce.foundation.ConfigurableEntity`. During inspection of the Javadoc for `Schema` and `ConfigurableEntity`, the reader may see the words “schema” and “scope” used interchangeably with “Property Set.” Figure 3-2 shows a simplified representation of property value retrieval through a `ConfigurableEntity`. For the `ConfigurableEntity`, the value of `backgroundColor` for `portalB` has been overridden. The value of `backgroundColor` for `portalA` has not. Therefore, when `backgroundColor` is requested for the `portalB` property set, the overridden value, red, will be returned. When `backgroundColor` is requested for the `portalA` property set, the property set default value, white, will be returned.

Figure 3-2 `backgroundColor` Property Retrieval

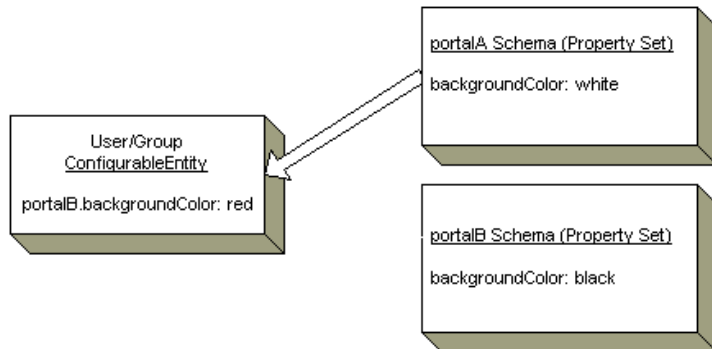


Figure 3-3 shows another simple example of `backgroundColor` property retrieval to demonstrate the notion of an explicit successor. A second `ConfigurableEntity` can be specified in the `ConfigurableEntity.getProperty()` API that acts as a “backup” place to look for a particular property value. This second `ConfigurableEntity` is

considered an explicit property successor. In this example, a particular group is used as an explicit successor, and the value for portalA's background color, green, is "inherited" from this successor.

Figure 3-3 Explicit Successor backgroundColor Property Retrieval.

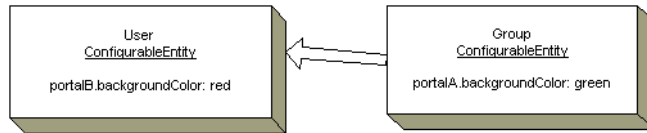
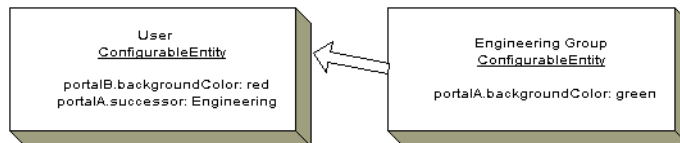


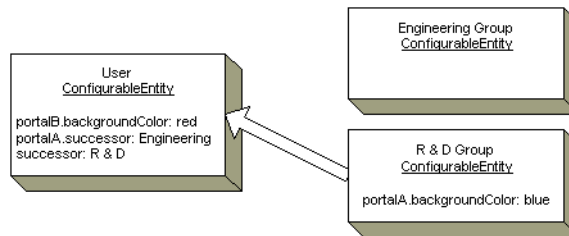
Figure 3-4 provides an example of an implicit successor. An implicit successor is a successor tied to a particular Property Set. In this case, the user does not have a value for portalA.backgroundColor, and no explicit successor is provided in the getProperty() call. However, the group has already been associated with the user as its successor for the portalA Property Set. Again, the user "inherits" the property value, green, from the group.

Figure 3-4 Property Inheritance Through Property Set-related Successor.



There also exists the notion of a default successor, which can be searched after an explicit successor and a Property Set-related successor have failed to return a value for the property. Figure 3-5 shows such a case. In this example, the Property Set-related successor cannot produce the necessary property value for backgroundColor in portalA, so the value must be retrieved from the default successor.

Figure 3-5 Figure 5. Property Inheritance Through a Default Successor.



Keep in mind that these examples have been considerably simplified for brevity and to easily explain relevant concepts. More details of `ConfigurableEntity` property inheritance are available in the topic “Users and Groups” in the chapter [Creating and Managing Users](#).

Using the Property Set Management Tool



Property Sets

create

Click a title link to edit and delete property sets and properties, or click create to add new property sets.

The Property Set Management tools allow you to create and manage sets of typed properties. Property Sets may be defined to describe user and group, session, request, and content properties.

Creating Property Sets

To create a property set:

1. On the Administration Tools Home page, click the Property Set Management icon. The Property Set Management Home page appears.
2. Click Create in the Property Sets banner. The Create Property Set page appears.
To enter a new property set:
 - a. Enter the name of the new property set in the Name field.
 - b. Enter a description of the new property set in the Description field.
 - c. Leave the Copy Properties From default as *Don't copy properties*.
 - d. From the Property Set Type drop-down list, select a property set type.To copy properties from an existing property set into the new one:
 - a. Enter the name of the new property set in the Name field.
 - b. Enter a description of the new property set in the Description field.
 - c. From the Copy Properties From drop-down list, select the property set containing the properties you want copied.
3. Click Create to create the property set.
4. Click Back to return to the Property Set Management Home Page.

Note: At any time, you can click Back to return to the Property Set Management Home Page without saving the property set.

Creating Properties Within a Property Set

To create properties within a property set:

1. On the Administration Tools Home Page, click the Property Set Management icon. The Property Set Management Home Page appears.
2. From the Property Set list, click the title link for the property set to which you will add a property. The Property Set view page appears.
3. Click Create on the Properties bar. The Create Properties page appears.

3 Creating and Managing Property Sets

Create Properties

Enter the appropriate information and click create.

Property Name: *

Description: *

Type:

Selection Option:

Creation Category:

back create

- a. Enter the property name in the Property Name field.
- b. Enter a description of the new property in the Description field.
- c. Select the type from the Type drop-down list box.
- d. Select option (single, multiple) from the Selection Option drop-down list box.

Note: The single option refers to those properties having only one option (for example, Property: Color, Attribute: red). The multiple option refers to those properties having multiple options (for example, Property: Colors, Attributes: red, green, blue, and so on).

- e. Select the creation of category (Restricted, Unrestricted) from the Creation Category drop-down box.

Note: Restricted categories refer to values that are selected via a list, radio buttons, check boxes, and so on. Unrestricted categories refer to instances in which users populate a form field.

4. Click Create.
5. Click Back to return to the Property Set view.

Setting Up the Property Default Value

Notes: Different steps are required for setting up default values, given your option/category selection.

To set up the property default value for single/restricted categories:

1. From Property Set view, click Edit on the appropriate Property Description bar.

2. Click Edit on the Properties Values bar.
3. Enter a new value to the property in the New Value field.
4. Click Create. The new value appears in the Values matrix at the bottom of the page.
5. Indicate the default value(s) by selecting the appropriate radio button.
6. Click Create.

To set up the property default value for single/unrestricted categories:

1. From Property Set view, click Edit on the appropriate Property Set Description bar.
2. Click Edit on the Properties Values bar.
3. Enter a new value to the property in the New Value field.
4. Click Create.

To set up the property default value for *multiple/restricted* categories:

1. From Property Set view, click Edit on the appropriate Property Set Description bar.
2. Click Edit on the Properties Values bar.
3. Enter a new value to the property in the New Value field.
4. Click Create. The new value appears in the Values matrix at the bottom of the page.
5. Indicate the default value(s) by selecting the appropriate radio button(s).
6. Click Create.

To set up the property default value for *multiple/unrestricted* categories:

1. From Property Set view, click Edit on the appropriate Property Set Description bar.
2. Click Edit on the Properties Values bar.
3. Enter a new value to the property in the New Value field.
4. Click Save.

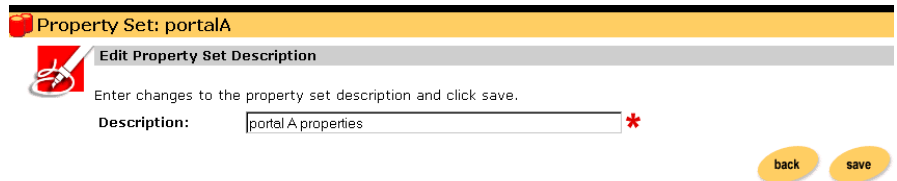
Editing Property Sets

To edit a property set:

1. On the Administration Tools Home Page, click the Property Set Management icon. The Property Set Management home page appears.
2. Click the appropriate title link from the Property Sets list. The Property Set view page appears.

To edit the Property Set Description:

1. Click Edit on the Property Set Description bar. The Edit Property Set page appears.



Property Set: portalA

Edit Property Set Description

Enter changes to the property set description and click save.

Description: *

back save

2. Enter the new description in the Description field.
3. Click Save to save changes. The general Property Set view appears with the new information. Alternately, click Back to return to Property Set view page without saving your changes.

Editing Properties Within a Property Set

To edit properties within a property set:

1. On the Administration Tools Home Page, click the Property Set Management icon. The Property Set Management home page appears.
2. Click the appropriate title link from the Property Sets list. The Property Set view page appears.
3. Click Edit on the appropriate property bar. The specific Property view page appears, containing information specific to the property you wish to edit.
4. Click Edit on the appropriate Description or Property Values bar. The Edit Property page appears.

5. Enter changes in the field(s) provided.
6. Click Save. The specific Property view returns. Alternatively, click Back. The specific Property view appears and your changes are not saved.

Deleting Property Sets

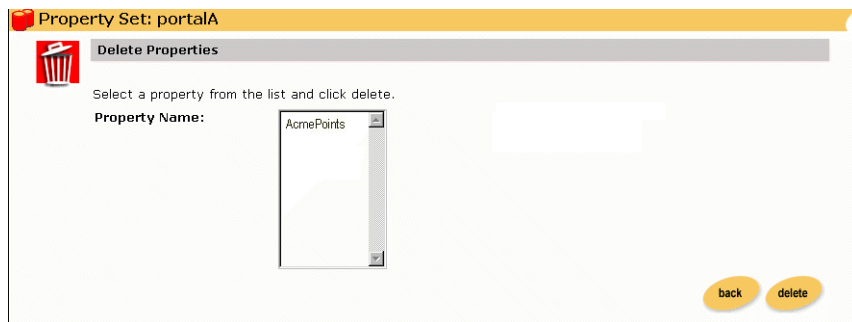
To delete a property set:

1. On the Administration Tools Home Page, click the Property Set Management icon. The Property Set Management home page appears.
2. Click the X to the right of the appropriate title link from the Property Set list.
3. Click OK to confirm the deletion.

Deleting Properties

To delete properties:

1. On the Administration Tools Home Page, click the Property Set Management icon. The Property Set Management home page appears.
2. Select the appropriate title link from the Property Sets list. The general Property Set view appears.
3. Click Delete on the Properties bar. The Delete Properties page appears.



3 *Creating and Managing Property Sets*

4. Select a property from the Property Name list.
5. Click Delete.
6. Click OK to confirm the deletion. The specific Property view returns.
Alternatively, click Back. The Property view appears and the property is not deleted.

4 Creating and Managing Users

This chapter discusses how User Management combines enterprise data about users with profile data that is used to personalize the users' view of the application.

This topic includes the following sections:

- Overview of User Management
- Users and Groups
- Unified User Profiles
- Using WebLogic Realms
- Anonymous User Profiles
- Privacy Statement
- User Manager
- Using the User Management Tool
 - Creating Groups
 - Deleting Groups
 - Adding Users to Groups
 - Removing Users from Groups
 - Editing Group Property Values
 - Creating Users
 - Editing User Property Values

- Deleting Users
- Creating Unified Profile Types
- Editing Unified Profile Types
- Deleting Unified Profile Types
- Using Other Realms
 - Registering User Attributes for Retrieval from LDAP
 - Unregistering User Attributes for Retrieval from LDAP
 - Registering Group Attributes for Retrieval from LDAP
 - Unregistering Group Attributes for Retrieval from LDAP
 - Viewing LDAP Configuration Settings
 - Selecting Groups for Use in the Personalization Server from the Realm
 - Mapping Realm Groups to the Personalization Server
 - Deleting Groups from Your Database

Note: Throughout this chapter, the environment variable `WL_COMMERCE_HOME` is used to indicate the directory in which you installed the WebLogic Commerce Server 3.1 and WebLogic Personalization Server 3.1 software.

Overview of User Management

The User Management system is a set of JSP tags, EJBs, and tools that facilitate the creation and persistence of user and group profile properties. It provides access to user profile information within a larger personalization server solution. In addition, the User Management system provides user-authentication mechanisms and user-to-group associations.

The User Management system responsibilities include:

- **User Authentication**—the user management system is used to authenticate a user against a persistent set of authentication information (typically a username-password combination).

- **User/Group Association Management**—the association of a user with one or more groups can play an essential role in determining user profile information pertinent to the user’s session. The user management system can either provide a default schema for user-group information persistence, or interface with existing user databases via standardized interfaces (for example, LDAP) or customized connectors.
- **User Profile Management**—the user management system constructs the user profile from persisted user and group attributes. User attributes can range from statically-defined properties, such as a user’s social security number, to dynamically-created and persisted properties, such as Web site tracking information for a particular user, or user preferences entered from a standard input screen. The user management system facilitates the creation and persistence of user profile properties.

Users and Groups

The two primary components employed by the Personalization Server’s User Management system are the User and Group, which extend ConfigurableEntity. It is from these components that User, and Group, and Unified User Profile functionality stems. User and Group components are also referred to as “user profiles” and “group profiles.”

The fully qualified name of each object is as follows:

- User: `com.beasys.commerce.axiom.contact.User`
- Group: `com.beasys.commerce.axiom.contact.Group`
- ConfigurableEntity:
`com.beasys.commerce.foundation.ConfigurableEntity`

The User Management system works in conjunction with the WebLogic Server’s security realm. In this arrangement, the security realm provides a list of users and groups, group membership information, and authentication. The User Management system uses the security realm to authenticate users and to know which users and groups exist and are valid, and which users are in a group. With this information from the security realm, it is possible for the User Management system to accomplish its

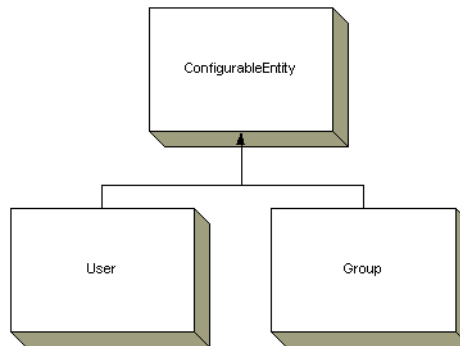
primary duties: creating, retrieving, and managing user and group profiles complete with property data. A default security realm (User Management RDBMSRealm) is provided by the Personalization Server as part of its “out-of-the-box” configuration.

Property data can be anything that is relevant to a user or group profile in the context of your personalized application. Things like age, gender, and favorite genres of music could all be property data. Things like department, position, and office location could also be property data. Much more is explained later about the actual and possible implementation details of handling property data in user and group profiles.

Group hierarchies permit property inheritance. For example, if a user profile does not yet have a “backgroundColor” property value, then the backgroundColor property value might be inherited from an “engineering” group. Groups may have only one or no parent group. As will be discussed later in this chapter, even if a realm for a third-party data store (for example, LDAP server) is used to access users and groups, any arbitrary group hierarchy may be configured for personalization purposes (property inheritance) via the User Management tools.

Profile functionality for both the User and Group components is inherited from the ConfigurableEntity implementation. Figure 4-1 shows a simplified representation of the User-Group-ConfigurableEntity relationship.

Figure 4-1 The User-Group-ConfigurableEntity Relationship



Unified User Profiles

In the BEA WebLogic Personalization Server, system users are represented by user profiles. A user profile provides an ID for a user and access to the properties of a user, such as age or e-mail address. Property values can be single-valued or multi-valued, and are requested via a *getProperty()* method which takes a property name as a key.

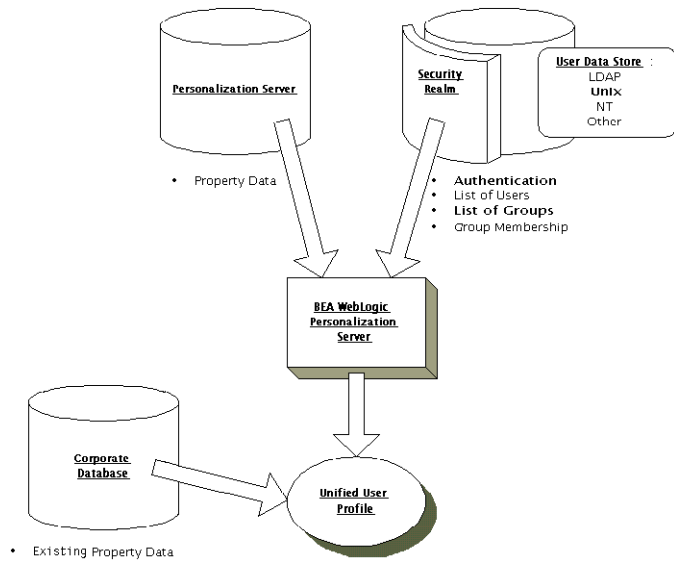
An advantage of the user profile is that it can be extended and customized to retrieve user information from an existing data source. For example, the user profile that ships with the Personalization Server can combine user properties from the Personalization Server database with user properties from an LDAP server into a single user profile for use within an application. Developers and system users need not worry about the different underlying data sources. To them there is just one place to go for user information – the user profile.

The Unified User Profile (UUP) is the name used to describe this aggregation of properties from an existing data source and the Personalization Server database tables into a single, customized user profile. More specifically, a UUP marries existing user/customer data by extending BEA's User component. By installing the Personalization Server's database tables into the existing database instance and extending the provided `com.beasys.commerce.axiom.contact.User` implementation, developers can quickly create a customized UUP that retrieves and stores properties from/to the existing database. This powerful flexibility is desirable because it allows access to existing data without requiring data migration or disrupting existing applications that also use the data. Conversely, if it is more desirable to migrate existing data into a separate Personalization Server database instance, this is also possible.

Configuration 1

Users and groups exist in some type of data store already, such as an LDAP directory. Existing user property data must be incorporated into the Unified User Profile as shown in Figure 4-2.

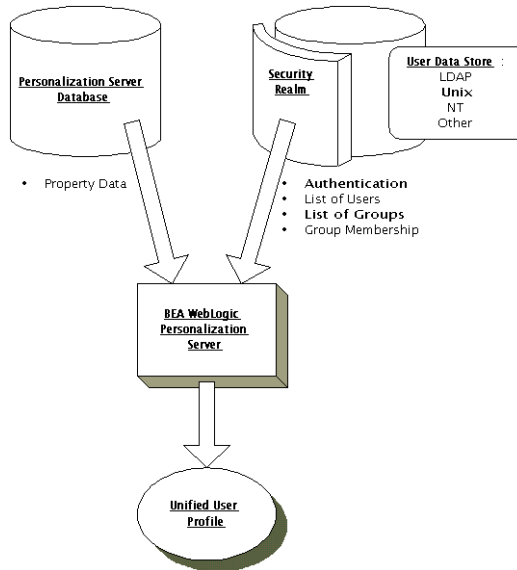
Figure 4-2 Configuration 1



Configuration 2

Users and groups already exist in a data store such as an LDAP directory. No existing user or group data must be incorporated into the Unified User Profile. All user and group property data is stored in the Personalization Server's database tables as shown in Figure 4-3.

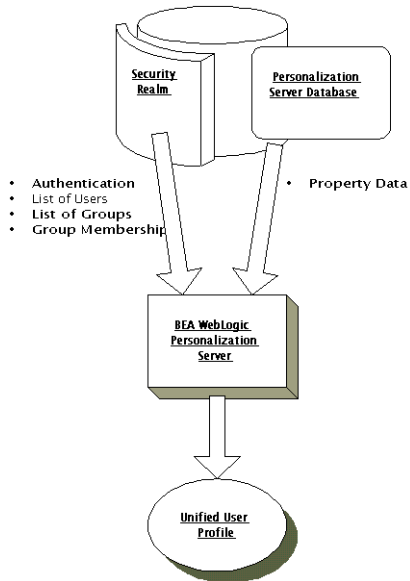
Figure 4-3 Configuration 2



Configuration 3

There is no existing store of users and groups. The Personalization Server's database tables contain all user and group data as shown in Figure 4-4.

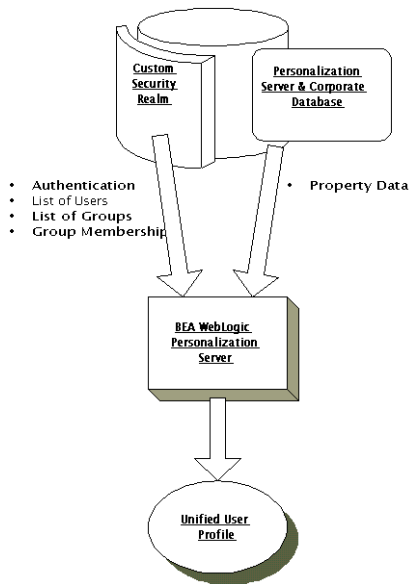
Figure 4-4 Possible Configuration 3



Configuration 4

User, group, and property data are in an existing database. Existing user property data must be incorporated into the Unified User Profile. A custom realm must be created in order to use the existing users and groups with the Personalization Server as shown in Figure 4-5.

Figure 4-5 Possible Configuration 4



The UnifiedUser example, found at [%WL_COMMERCE_HOME%/server/public_html/examples/unifieduserprofile/index.htm](#), demonstrates a fictitious company's use of the UUP to take advantage of existing customer data. The UnifiedUser extends `com.beasys.commerce.axiom.contact.User` and retrieves data from a pre-existing database. If you have existing user information that you wish to leverage in your application, it is recommended that you study this example. The UnifiedUser

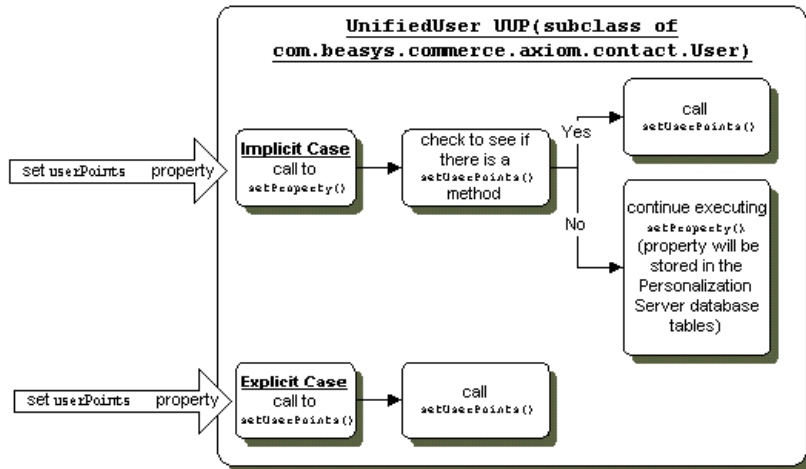
shows how, with relative ease, you can create a customized UUP that suits your application's persistence needs. The following table explains exactly what must be extended in order to create your own custom UUP.

Object	Must Extend
UUP Primary Key	<code>com.beasys.commerce.axiom.contact.UserPk--</code> with no key fields added.
UUP EJB Interface	<code>com.beasys.commerce.axiom.contact.User</code>
UUP EJB Implementation	<code>com.beasys.commerce.axiom.contact.UserImpl</code>

The fact that UUPs are `ConfigurableEntities` means that user profiles have the notion of setting and getting a property explicitly or implicitly. Explicitly setting a property means calling a setter method for a property directly. Implicitly setting a property means setting a property via the `setProperty()` method where no explicit setter method is available. For example, if a UUP contains a “userPoints” property, calling `setUserPoints()` directly would explicitly set the `userPoints` property, while calling `setProperty()` with the “userPoints” key would implicitly set the `userPoints` property. When it is called, `setProperty()` will first look for a `setUserPoints()` setter method to call in the user profile. If such a setter method exists, this method is called and is responsible for setting the property and doing whatever else is necessary regarding that property's change in value. Ultimately it is the UUP implementation's responsibility to persist explicitly-set property values—even if they are implicitly called via `setProperty()`. `ConfigurableEntity` only handles persisting implicitly set properties where no explicit setter method exists.

Figure 4-6 diagrams both an explicit and implicit call to `setUserPoints()`. In both cases, it is the UUP bean's responsibility to handle storing the `userPoints` value. If no `setUserPoints()` method had existed in the UUP bean, the `ConfigurableEntity` implementation would have handled storing the `userPoints` value.

Figure 4-6 Implicit and Explicit Calls to Set the userPoints Property



This notion of implicitly and explicitly setting properties allows for additional flexibility in UUP implementation. If any special logic needs to happen during the setting or getting of a property, such as the recalculation of some other value, it can conveniently be done in a setter or getter method for that property. Functionality external to the UUP can always count on having a `setProperty()` method and a `getProperty()` method for access to properties, eliminating any need to know whether a property has its own setter or getter. For example, the `<um:getProperty>` JSP tag can always retrieve the `userPoints` property value even if a `getUserPoints()` method is the only way provided by the UUP to retrieve `userPoints`. This is because the UUP's `getProperty()` method will first check to see if it has a `getUserPoints()` method before checking elsewhere. Properties that have an explicit `set<PropertyName>()` and `get<PropertyName>()` method are referred to as “explicit properties,” while properties that can only be set through a call to `setProperty()` are referred to as “implicit properties.”

When implementing a custom UUP EJB, you only need to worry about implementing explicit getter and setter methods for the explicit properties you want the UUP to have. The implementations of these setters and getters then do whatever is necessary to set and retrieve the property values in the existing datastore.

There are a few important things to be aware of when creating a custom UUP. The `get<PropertyName>()`, `set<PropertyName>()` convention must be followed for all explicit property setting and getting in a UUP. This means if you have a UUP with an explicit `userPoints` property, you must provide an explicit `getUserPoints()`

method—`retrieveUserPoints()` would not work. Similarly, setting `userPoints` must be done with a `setUserPoints()` method. This is because the `getProperty()` and `setProperty()` methods look for getters and setters that follow this convention when getting and setting properties via implicit calls. Overriding `setProperty()` or `getProperty()` is not permitted—all getting and setting of explicit properties must be done through getter and setter methods. Explicit getters and setters must take and return objects—primitives such as `long` and `float` must be wrapped in `java.lang.Long` and `java.lang.Float` objects to be compatible with `ConfigurableEntity`'s `getProperty()` and `setProperty()` methods.

Also, if you provide a getter method, it is a good idea to also provide a setter method and vice versa. This is because you can never predict when someone will try to set or get a property. For example, let's say you provide a getter that retrieves a property from a database table but no corresponding setter. If `setProperty()` is called for that property it will be stored in a Personalization Server table. This is messy because you have the value being retrieved from one place and set in another. The next time the property is retrieved, it would have its original value—not the value that was set. If you want to provide a read-only property, you should implement an empty setter method.

The definition of `ConfigurableEntity`'s `getProperty()` method is as follows:

```
public Object getProperty(String propertySet,
                          String propertyName,
                          ConfigurableEntity explicitSuccessor,
                          Object defaultValue);
```

The `getProperty()` method searches for properties in different places in a specific order which is important to understand. For example, if a property is not found for a `User`, perhaps a `Group` should be queried for the value. In this case the `User` would inherit the property value from a `Group`. In `ConfigurableEntity` terms, the `Group` would be the `User`'s "successor." If a property is not found in a `ConfigurableEntity`, then the `ConfigurableEntity`'s successor is queried for the value. This way `ConfigurableEntities` can inherit and override values from a parent entity. Successors can be implicit or explicit. An implicit successor is a `ConfigurableEntity`'s default successor or a successor that is set for a specific `Property Set`. An explicit successor is a `ConfigurableEntity` that is passed as a parameter to the `getProperty()` method. Following is the order of the `getProperty()` property search as it exists in `ConfigurableEntity`, and hence the `User` and `Group` objects as well as any `UUP` objects:

1. Look for an explicit `getter` method for that property.
2. Look in the entity for the property for the specified `Property Set`.
3. Look in the entity for the property in the default (null) `Property Set`.

4. Look in the entity for the property in the Reserved Property Set (for properties from LDAP if using the LDAPRealm).
Note: Properties to be retrieved from LDAP must be registered as LDAP attributes. See [“Registering User Attributes for Retrieval from LDAP” on page 4-40](#).
5. Look for the property in the entity’s explicit successor (if specified).
6. Look for the property in the entity’s successor for the specified Property Set.
7. Look for the property in the entity’s default successor.
8. Look for a default value as defined in the Property Set if the Property Set is specified (not null).
9. Return the default value passed into the `getProperty()` method.

The definition of `ConfigurableEntity`’s `setProperty()` method is as follows:

```
public Object setProperty(String propertyName,
                        String propertyValue,
                        Object value);
```

This method has a few details that are also important to understand. If `setProperty()` is used to set a property for a Property Set that is inconsistent with the property set’s definition, an exception is thrown. For example, suppose we have defined a “UnifiedUserExample” Property Set that has a `userPoints` property of type `Integer`. If someone tries to set the `userPoints` property for the “UnifiedUserExample” Property Set to be “foo,” an exception would be thrown because `userPoints` is defined as being of type `Integer` and “foo” is text. Similarly, setting a `Boolean` property value to “bar” would result in an exception because `Boolean` values are restricted to `Boolean` objects.

If `setProperty()` is called and `null` is passed for the Property Set, the property value is set in the `null` Property Set—referred to as the default Property Set. As described previously in the search order of `getProperty()`, the default property set is searched before looking for the property value in the “Reserved” Property Set and then a successor.

The “Reserved” Property Set is a read-only Property Set that is used to hold property values from an external datastore. The only time the “Reserved” Property Set is currently used in the Personalization Server is when properties are retrieved from an LDAP directory. Attempting to set a property in the “Reserved” Property Set will result in an exception being thrown. Properties in the “Reserved” Property Set and the

Reserved Property Set itself are not editable via the User Management tools. The User Management tools allow the specification of attributes to be retrieved from an LDAP server for users and groups. Only these attributes will be retrieved at runtime.

Properties can be set via `setProperty()` with a Property Set specified that does not exist. This is allowed, but strongly discouraged. When this is done, a Property Set is not created “on-the-fly” for the specified Property Set name. Rather, the specified Property Set name serves only as a namespace for the property. Similarly, it is allowed but strongly discouraged to set a property via `setProperty()` for an existing Property Set specifying a property that does not exist for that Property Set. Properties set in either of these ways are not editable through the User Management tools, but properties in the “null” (“default”) property set are editable from the tools.

A couple of additional points about `getProperty()` and `setProperty()` that are worth mentioning are as follow:

- `getProperty()` returns a `java.lang.Long` object if `setProperty()` is called passing a `java.lang.Integer` object value. Code retrieving such a property should be written as follows:

```
Object value      = myUser.getProperty("my_property_set",
                                       "my_integer_property",
                                       null,
                                       null);
Number tempNumber = (Number) value;
int    realValue  = tempNumber.intValue();
```

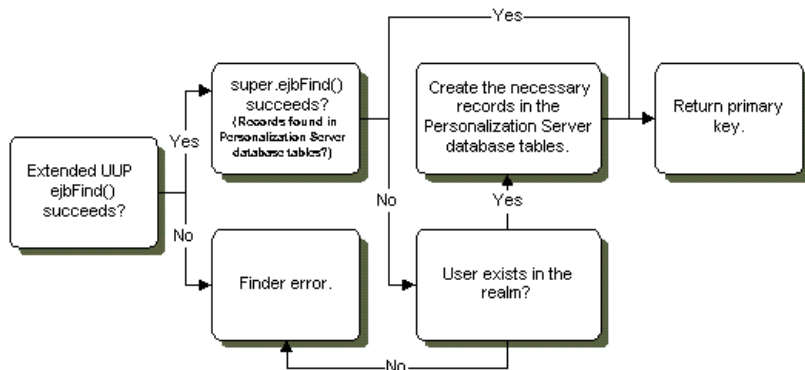
- `getProperty()` returns a `java.lang.Double` object if `setProperty()` is called with a `java.lang.Float` object. Code retrieving such a property should be written as follows:

```
Object value      = myUser.getProperty("my_property_set",
                                       "my_float_property",
                                       null,
                                       null);
Number tempNumber = (Number) value;
float  realValue  = tempNumber.floatValue();
```

The `com.beasys.commerce.axiom.contact.User` object offers functionality for EJB find operations that makes integrating a UUP with the Personalization Server easy. Once a UUP's `ejbFind()` finds records in the existing data store, the call to `super.ejbFind()`--the User object `ejbFind()`-- will create the necessary records for the UUP in the Personalization Server tables if they do not yet exist and the following condition is met: If the User object `ejbFind()` fails, it checks the underlying security realm to see if the username corresponds to a valid user. If so, User's `ejbFind()`

creates the necessary records, thereby eliminating finder errors and the need to spend time initially migrating user data into the Personalization Server's User database tables (Figure 4-7).

Figure 4-7 Flow During an `ejbFind()` Operation



If your configuration is such that the realm cannot verify the existence of the user, but the user must be created, it is the responsibility of your EJB to create the superclass records if they are not found initially. The UnifiedUser example code demonstrates such a situation. Please refer to the `ejbFindByPrimaryKey()` method in the file `UnifiedUserBean.java`.

Six entries are required in the `ejb-jar.xml` file used when creating the unified user profile bean's descriptor. There entries are:

1. JNDIHomeName

This environment entry is not to be confused with the actual JNDI lookup name of the extended EJB. Rather, it is used to relate profile entries for the UUP EJB with those of `com.beasys.commerce.axiom.contact.User`. The value must always be:

```
com.beasys.commerce.axiom.contact.User
```

Exact entry:

```
<env-entry>
  <env-entry-name>JNDIHomeName</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>
    com.beasys.commerce.axiom.contact.User
```

```
</env-entry-value>  
</env-entry>
```

2. SchemaGroupName

This environment entry is used to configure the EJB to pull property values from a particular classification of Property Sets. The value must always be:

USER

Exact entry:

```
<env-entry>  
  <env-entry-name>SchemaGroupName</env-entry-name>  
  <env-entry-type>java.lang.String</env-entry-type>  
  <env-entry-value>USER</env-entry-value>  
</env-entry>
```

3. SmartBMPClass

This environment entry specifies which SmartBMP class to use when creating, refreshing, updating, and removing the EJB. If you have created a SmartBMP for your class which extends

`com.beasys.commerce.axiom.contact.UserSmartBMP`, use the classname of your SmartBMP for this entry. If you do not use a particular SmartBMP with your class, use `com.beasys.commerce.axiom.contact.UserSmartBMP` as the value.

Sample entry:

```
<env-entry>  
  <env-entry-name>SmartBMPClass</env-entry-name>  
  <env-entry-type>java.lang.String</env-entry-type>  
  <env-entry-value>  
    com.beasys.commerce.axiom.contact.UserSmartBMP  
  </env-entry-value>  
</env-entry>
```

4. EntityPropertyManagerHome

This environment entry specifies which `EntityPropertyManager` bean to use when accessing user and group properties. If using the LDAP configuration (security realm is the LDAP realm), the entry must be as follows.

Exact Entry:

```
<env-entry>  
  <env-entry-name>EntityPropertyManagerHome</env-entry-name>  
  <env-entry-type>java.lang.String</env-entry-type>  
  <env-entry-value>
```

```

    com.beasys.commerce.foundation.property.EntityPropertyAggregator
  </env-entry-value>
</env-entry>

```

For any other configuration the `EntityPropertyManagerHome` entry should be specified as follows.

Exact Entry:

```

<env-entry>
  <env-entry-name>EntityPropertyManagerHome</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>
    com.beasys.commerce.foundation.property.EntityPropertyManager
  </env-entry-value>
</env-entry>

```

The contents of the `ejb-jar.xml` file shipped with the UnifiedUser example are shown below. Note that this bean was not paired with its own SmartBMP implementation derived from UserSmartBMP.

5. PersistenceHelperPlugin

This entry specifies which persistence helper class should be used by the BMP. If the standard UserSmartBMP is being used, the value should be “`com.beasys.commerce.foundation.plugin.bmp.BMPPersistenceHelperPlugin`”.

Exact Entry:

```

<env-entry>
  <env-entry-name>PersistenceHelperPlugin</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>com.beasys.commerce.foundation.plugin.bmp.
BMPPersistenceHelperPlugin</env-entry-value>
</env-entry>

```

6. UnifiedProfileType

This entry specifies the type of Unified Profile that this class belongs to. It is necessary to transparently create, edit, and delete UUP users through the admin tools. In the Unified User Example, the value is “Unified Profile Example”.

Exact Entry:

```

<env-entry>
  <env-entry-name>UnifiedProfileType</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>Unified Profile Example</env-entry-value>
</env-entry>

```

4 *Creating and Managing Users*

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>examples.usermgmt.UnifiedUser</ejb-name>
      <home>examples.usermgmt.UnifiedUserHome</home>
      <remote>examples.usermgmt.UnifiedUser</remote>
      <ejb-class>examples.usermgmt.UnifiedUserBean</ejb-class>
      <persistence-type>Bean</persistence-type>
      <prim-key-class>examples.usermgmt.UnifiedUserPk</prim-key-class>
      <reentrant>False</reentrant>
    <env-entry>
      <env-entry-name>JNDIHomeName</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>

    <env-entry-value>com.beasys.commerce.axiom.contact.User</env-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>SchemaGroupName</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>USER</env-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>SmartBMPClass</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>

    <env-entry-
value>com.beasys.commerce.axiom.contact.UserSmartBMP</env-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>EntityPropertyManagerHome</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>

    <env-entry-
value>com.beasys.commerce.foundation.property.EntityPropertyAggregator</env-ent
ry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>PersistenceHelperPlugin</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>

    <env-entry-
value>com.beasys.commerce.foundation.plugin.bmp.BMPPersistenceHelperPlugin</env
-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>UnifiedProfileType</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>Unified Profile Example</env-entry-value>
    </env-entry>
```

```

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

  </entity>
</enterprise-beans>
<assembly-descriptor></assembly-descriptor>
</ejb-jar>

```

Additionally, the following entry must be added to the `weblogic-ejb-jar.xml` file of the UUP so that it can access the `commercePool` database connection pool:

```

<weblogic-enterprise-bean>
  [...]

  <reference-descriptor>
    <resource-description>
      <res-ref-name>jdbc/commercePool</res-ref-name>
      <jndi-name>weblogic.jdbc.jts.commercePool</jndi-name>
    </resource-description>
  </reference-descriptor>
  [...]
</weblogic-enterprise-bean>

```

The last step in completing a custom UUP requires the UUP to be registered with the Personalization Server through the User Management tools. In order to register the UUP, the User Management tools require the following:

Item	Description
Profile Type Name	Arbitrary name that is later used to refer to the profile type through the User Management system's <code><um:getProfile></code> JSP extension tag.
Profile Home Class	The home class of the new profile type.
Profile Remote Interface	The remote interface of the new profile type.
Profile Primary Key Class	The primary key class of the new profile type.
Profile JNDI Name	The JNDI lookup name of the new profile type.

By registering the UUP with the Personalization Server, it becomes possible to ask for the new profile type with the `<um:getProfile>` JSP tag:

```
<um:getProfile profileType="UnifiedUserExample"  
profileKey="<%=username%>" />
```

It is then possible to use the `<um:getProperty>` and `<um:setProperty>` JSP tags with the UUP.

Using WebLogic Realms

A realm is a Java class that provides access to a store of Users, Groups, ACLs (Access Control Lists), and related services. WebLogic Server uses a realm as a service, calling into the realm to retrieve Users, Groups, and ACLs as Java objects. WebLogic Server provides realms that access the WebLogic Server properties file, Windows NT, or UNIX networks, and LDAP servers for user, group, and ACL information. The WebLogic Personalization Server provides an additional RDBMSRealm which uses its own database tables containing user and group information as an out-of-the-box option. It is also possible to create your own realm if your situation requires accessing a datastore not supported by WebLogic Server.

The WebLogic Personalization Server must have access to a realm to retrieve information about users and groups, determine a group's members, and authenticate users. By depending on realms, the Personalization Server can use existing stores of user and group information, allowing that information to remain in place. For instance, if you already have users and groups defined in an LDAP directory, they can be accessed by the Personalization Server through the LDAPRealm without requiring any redundant data entry.

If you are using the Personalization Server without an external data store of user and group information, then that information will be stored in the Personalization Server's database tables. In this case, the

`com.beasys.commerce.axiom.contact.security.RDBMSRealm` must be used to access user and group information from the Personalization Server tables. For this configuration to work, the appropriate realm properties for your database type must exist in the `commerce.properties` file. Make sure the following properties are set:

In the BEA WebLogic Personalization Server's `commerce.properties` file:

If using Oracle Thin Drivers (Oracle 8.0.5, Oracle 8.1.5):

```
commerce.usermgmt.RDBMSRealm.driver=oracle.jdbc.driver.OracleDriver
commerce.usermgmt.RDBMSRealm.dbUrl=\
    jdbc:oracle:thin:@<machine name>:<port number>:<database
instance>
commerce.usermgmt.RDBMSRealm.dbUser=<database user>
commerce.usermgmt.RDBMSRealm.dbPassword=<database user's password>
```

If using the WebLogic Oracle OCI Driver:

```
commerce.usermgmt.RDBMSRealm.driver=weblogic.jdbc.oci.Driver
commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:weblogic:oracle
commerce.usermgmt.RDBMSRealm.dbServer=<machine name>
commerce.usermgmt.RDBMSRealm.dbUser=<database user>
commerce.usermgmt.RDBMSRealm.dbPassword=<database user's password>
```

If using Cloudscape:

```
commerce.usermgmt.RDBMSRealm.driver=COM.cloudscape.core.
JDBCdriver
commerce.usermgmt.RDBMSRealm.dbUrl=jdbc:cloudscape:Commerce;\
    create=true;autocommit=false
commerce.usermgmt.RDBMSRealm.dbUser=none
commerce.usermgmt.RDBMSRealm.dbPassword=none
```

In the BEA WebLogic Server's weblogic.properties file:

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

It is important to note that if a realm other than the Personalization Server's RDBMSRealm is being used, the Administration tools for creating users and groups become inaccessible. This is because adding users and groups and administering credentials must be done through tools provided by the external datastore.

For use within the Personalization Server, a realm must be a subclass of weblogic.security.acl.AbstractListableRealm. The WebLogic NTRealm, LDAPRealm, and UnixRealm are all subclasses of AbstractListableRealm.

Tools are provided that allow a properly-configured realm to be set up for use by the Personalization Server. The realm configuration tools allow you to choose which groups from the realm you wish to use in the Personalization Server, map group names that have changed in the realm to new group names, and clean up Personalization Server records that no longer correspond to valid realm users or groups.

Note: Changing the underlying realm can cause unpredictable behavior if the realm configuration tools are not immediately used to map and remove groups and clean up users as appropriate for the new realm.

In addition to user and group information, realms may also provide ACLs to determine an authenticated user's permissions within the system. An ACL guards an object or service in WebLogic Server. ACLs can guard servlets and JSP pages, JMS queues and topics, EJBs, JDBC connection pools, JNDI contexts, and ZAC packages. You can also create custom ACLs for use in your applications, and these ACLs will be supported by the Personalization Server.

An ACL holds a list of `AcIEntries`, each with a set of permissions for a user or group. A permission is an action that can be performed on the protected resource — for example, "execute," "lookup," "read," or "write." The exact permissions available depend on the type of resource the ACL protects. For example, a servlet requires "execute" permission, and a JMS queue requires "read" or "write" permission.

For more information on realms, including how to configure and administer realms, consult the WebLogic Server documentation for *Using WebLogic Realms and ACLs*. Also, for more information on implementing a custom realm, see the WebLogic Server documentation.

Anonymous User Profiles

Certain scenarios require an unidentified user to be able to use a system. While the unidentified user is using the system, you may need to have a profile for that user in order to set and get properties. For instance, a portal Web site might want to let new users tour the Web site and configure a few things before they actually have an official login name and password. The anonymous user profile allows for a user profile to be created for such a user. An anonymous user profile can be treated just like a user profile for a known user, but the anonymous user profile only lives for the life of the user session. If the session is terminated without capturing an identity for the user, any

profile information accumulated during the life of the anonymous user profile is lost. An anonymous user profile has no successor and will not retrieve default property values from a Property Set.

The anonymous user profile is available only through JSP tags. An anonymous profile is created when a `<um:setProperty>` or `<um:getProperty>` JSP tag is used before a `<um:getProfile>` tag has been called. If during a session a persistent user profile is created for the anonymous user, the `<um:createUser>` tag can be told to store the values from the anonymous profile into the new user profile. This is done with the `saveAnonymous` tag parameter set to `true`, as in `<um:createUser saveAnonymous="true">`. For more information on these tags, see the topic “User Management JSP tags” in [JSP Tag Library Reference](#) in the *WebLogic Personalization Server Developer’s Guide*.

For an example, see

`%WL_COMMERCE_HOME%/server/public_html/anonymousprofile/index.html`

Privacy Statement

The Platform for Privacy Preferences Project (P3P) is an emerging industry standard that is designed to provide an automated way to compare consumers’ privacy preferences with the privacy practices of the Web sites they visit. It lets Web sites express their privacy practices in a format that can be retrieved automatically and interpreted easily.

The P3P is a work-in-progress by the World Wide Web Consortium (W3C), a global group drawn from industry, academia, and privacy groups as well as public policy organizations. For more information about the World Wide Web Consortium’s ongoing P3P effort, visit the P3P site at <http://www.w3.org/P3P>.

Essentially, P3P compliance means that your Web site presents a privacy policy to the user. As put forth in the P3P specification, a privacy policy is a set of one or more privacy statements that describe what personal user data a Web site will retrieve, and how the data is to be used. The P3P specification currently defines three mechanisms by which a Web site’s privacy policy information can be presented to the end user:

- By publishing the policy reference file at a well-known URL.
For complete information, see the P3P specification, section 2.2.1.
http://www.w3.org/TR/P3P/#mechanism_ref

- By injecting a special header in each HTTP response served up by a the Web server. For complete information, see the P3P specification, section 2.2.2. http://www.w3.org/TR/P3P/#syntax_ext
- By using an embedded <link> tag in the body of an HTML page. For complete information, see the P3P specification, section 2.2.3. http://www.w3.org/TR/P3P/#syntax_link

BEA Systems applauds the efforts of the World Wide Web Consortium and other organizations around the world working to empower users to control the use of their personal information on the Web sites they visit. However, it is important to note that WebLogic Personalization Server does not in any way enforce P3P compliance—that option is left up to the Web site developer.

User Manager

The UserManager Session EJB provides user management functionality in a Personalization Server-specific context. Services provided by the UserManager include:

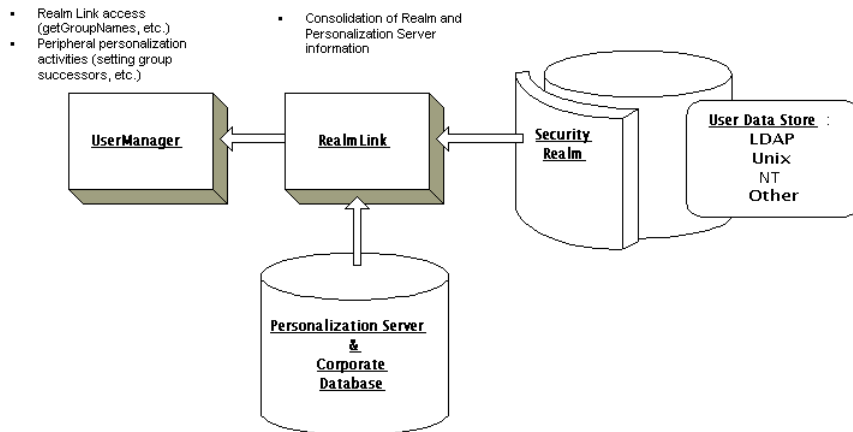
- Creating/removing users
- Creating/removing groups
- Adding users to groups/removing users from groups
- Adding groups to groups/removing groups from groups
- Retrieving usernames corresponding to a group
- Retrieving group names corresponding to a user
- Retrieving unique group and user IDs based on group/username
- Retrieving group/username based on unique ID
- Retrieving user/group objects based on name

For a complete list of UserManager services, please refer to the UserManager Javadoc.

Though it supplies the underlying functionality of the Group/User management JSP extension tags, the UserManager can be accessed directly. However, the UserManager is not intended for use outside the context of the Personalization Server. To emphasize this point, the general relationship between the UserManager and the security realm support mechanism will be briefly explained, followed by a few examples.

Figure 4-8 shows the relationship between the UserManger, the RealmLink, and the security realm. The RealmLink is used to ensure that realm query results are consistent with Personalization Server user and group data. The RealmLink is the only object aware of both the Personalization Server data, and the Realm user and group data. An example of RealmLink activity is the query for group names associated with a particular user. Since the user manager administration tools allow for group registration with the Personalization server, the RealmLink will only return group names for a particular user that exist in both the security realm and in the Personalization Server tables.

Figure 4-8 UserManger/RealmLink Cooperation



To ensure behavior consistent with Personalization Server purposes, the UserManager employs two primary strategies:

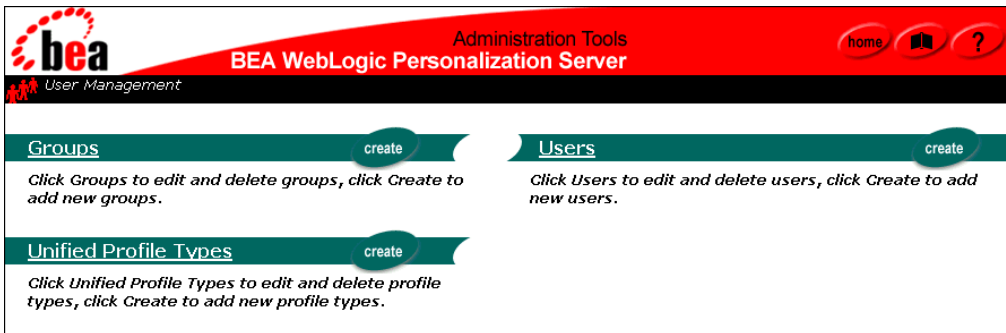
1. For certain operations, the UserManager qualifies the security realm being used before taking action. These operations can only be performed if the current security realm class is `com.beasys.commerce.axiom.contact.security.RDBMSRealm`. See [UserManager EJB in Javadoc](#) for details.

For example, the `createGroup()` method throws a `UserManagementException` if the out-of-the-box `RDBMSRealm` is not being used. The logic behind such an exception is that the `UserManager` is designed to work with the default Personalization database schema. If another realm is being used (for example, `WebLogic LDAPRealm`), it is assumed that the client has another means, besides the Personalization Server administration tools, that should be used for adding and removing groups and users to/from the realm.

2. For all operations, the `UserManager` works in conjunction with the `com.beasys.commerce.axiom.contact.security.RealmLink` class to ensure results consistent with both security realm and Personalization Server user and group data.

For example, the `getGroupNamesForUser()` method returns only group names which exist in the current security realm and which are registered with the Personalization Server via the Realm Configuration tools.

Using the User Management Tool



The User Management administration tools allow you to create and associate users and groups or to link to and use existing directories of users. A user or group may then be personalized by overriding property values as defined in the Property Set Management tool. The Unified Profile Types tool allows you to configure access through User Management tag libraries to your existing application EJBs.

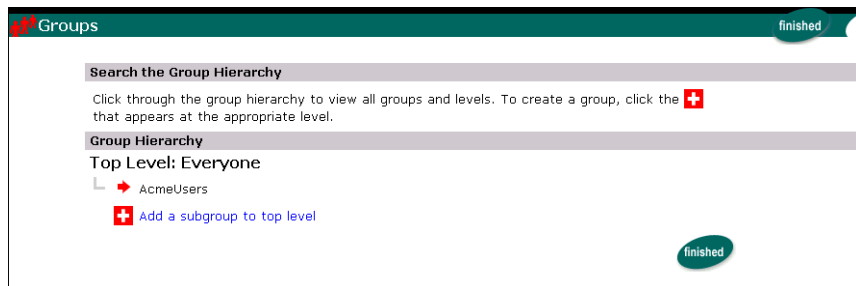
Note: If your system is configured for a third-party realm, the interface above would contain a Realm banner in addition to the ones presented and an LDAP banner if you are using the LDAP Realm. In addition, the Create buttons would not appear on the Users or Groups banners.

Creating Groups

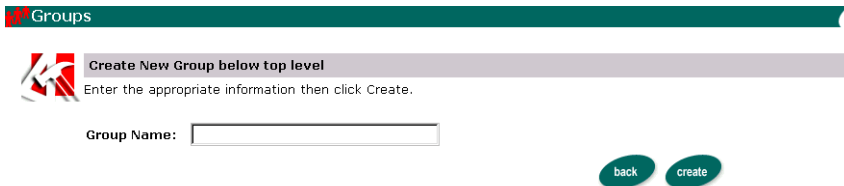
Note: The User Management tools do not allow the creation of a group called “everyone”, as this is a reserved WebLogic Server group name.

To create groups:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Create in the Groups banner. The Create a New Group page appears.



3. Within the Group Hierarchy tree view, expand the hierarchy as needed to display the add icon (+) at the level you wish to add the group. Click on the plus sign. The Create a Group page appears.
4. Enter the name of the new group in the Group Name field.
5. Click Create. A success or failure message appears.

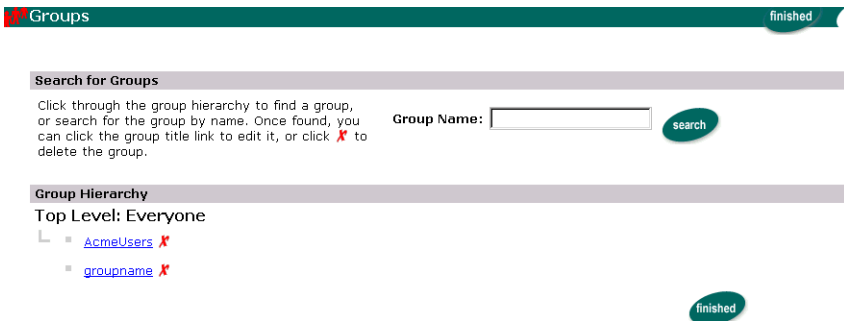


6. Click Back to return to the Group Administration tool or to enter another new group name (Step 4).

Deleting Groups

To delete groups:

1. On the Administration Tool Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Groups in the Groups banner. The Search for Groups tool appears.



- a. To locate the group to delete by name, enter the group name in the Group Name field, then click Search.

Note: The group name must be entered exactly.

- b. To locate the group to delete within the Group Hierarchy, navigate the Group Hierarchy tree view.

3. Click the X to the right of the group name. A confirmation box appears.
4. Select OK. The group is deleted.

Adding Users to Groups

To add users to groups:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Groups in the Groups banner. The Search for a Group page appears.

To locate the appropriate group, do one of the following:

- a. To locate the group by name, enter the group name in the Group Name field, then click Search.
 - b. To locate the group within the Group Hierarchy, navigate the Group Hierarchy tree view.
3. Select the group. The Group Properties view appears.
 4. Click the add/remove icon (+/-) at the bottom of the page. The Add/Remove Users tool appears.

Group: AcmeUsers



Add/Remove Group Search Results

Search for the user you want to add or remove from this group. The search results and current group users will appear at the bottom of the page. To add a user, select the user name and click the right arrow. To remove a user, select the user name and click the left arrow. You must click the "save" button to commit any changes to the group before performing a new search or leaving this page.

Username:

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Search Results:

Group Search Results:

To locate a user, do one of the following:

- To locate the user by name, enter the username in the Username field, then click Search. The search results appear at the bottom of the page.
 - To see a list of all users within an alphabetized category, click the appropriate letter corresponding to the first letter of the username. A list of users appear at the bottom of the page.
 - To see a list of all users in the database, use the wildcard feature. Enter a partial username immediately followed by an asterisk (*). The asterisk is a search return variable.
5. Select the username, or a group of names, from the Search Results field.

Search Results:

Group Search Results:

acme
auseername

6. Click the left-to-right directional arrow. The username(s) appears with the Group Users field.

7. Click Save.
8. Click Back to return to the Group Properties view.

Note: The search applies both list boxes.

Removing Users from Groups

To remove users from groups:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Groups in the Groups banner. The Search for Groups tool appears.

To locate the appropriate group, do one of the following:

- a. To locate the group by name, enter the group name in the Group Name field, then click Search.
 - b. To locate the group within the Group Hierarchy, navigate the Group Hierarchy tree view.
3. Select the group. The Group Properties view appears.
 4. Click the add/remove icon (+/-) at the bottom of the page. The Add/Remove Users tool appears.

To locate a user, do one of the following:

- a. To locate the user by name, enter the username in the Username field, then click Search. The search results appear at the bottom of the page.
 - b. To see a list of all users within an alphabetized category, click the appropriate letter corresponding to the first letter of the username. A list of users appear at the bottom of the page.
 - c. To see a list of all users in the database, use the wildcard feature. Enter a partial username immediately followed by an asterisk (*). The asterisk is a search return variable.
5. Select the username, or a group of usernames, from the Group Users field.

6. Click the right-to-left directional arrow. The username(s) is removed from the Group Users field and appears in Search Results.
7. Click Save.
8. Click Back to return to the Group Properties view.

Editing Group Property Values

To edit group property values:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Groups in the Groups banner. The Search for a Group page appears.

To locate the appropriate group, do one of the following:

- a. To locate the group by name, enter the group name in the Group Name field, then click Search.
 - b. To locate the group within the Group Hierarchy, navigate the Group Hierarchy tree view.
3. Select the group. The Group Properties view appears.
 4. Select or search for a property set to view for this group. For specific instructions on property set management, see [Chapter 3, “Creating and Managing Property Sets.”](#) The group’s default property values appear if no other property set has been accessed during the tools session.
 5. Click Search.
 6. Click Edit on the appropriate Property bar. The associated Edit Property Values page appears.
 7. Change the values on the Edit Property Values page.
 8. Click Save.
 9. Click Back to return to the Group Properties view.
 10. Return to step 4 and edit other properties as necessary.

Notes: Non-default Property sets and properties not configured through the Property Set Management tools are not editable here.

If you click the Reset button on the Property bar (instead of Edit as we did in step 6), the property is set to null for that user. This will have one of three results:

- First, if the property has a default value, the group will have that default value. Note that the default value is not copied into the group's settings. The group's value is just set to null so that the default value will be returned when `getProperty()` is called for that property. If the default value changes, calling `getProperty()` will return the new default value.
- Second, if the property is defined in a Property Set but does not have a default value, the user will have a null for that property.
- Third, if the property was dynamically defined (that is, it does not belong to a Property Set), resetting causes that property to be deleted.

Creating Users

To create users:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Create in the Users banner. The Create New Users page appears.

Users

Create New Users
Enter the appropriate information then click Create. New users will display in the list below.

Username: *

Password: *

Verify Password: *

User Type:
User
Unified Profile Example
WLCS_Customer

back create

3. Enter the username in the Username field.

Note: Limit usernames to 25 characters.

4. Enter the password associated with the Username in the Password field.
5. Re-enter the password provided in step 4 in the Verify Password field.

Note: Characters in password fields appear as asterisks.

6. From the User Type list, select a Unified Profile. The user will be an instance of this Unified Profile. This allows the system to access explicit properties in a Unified Profile type, and ensures proper data cleanup when the user is removed.
7. Click Create. The new user appears at the bottom of the page.
Alternatively, click Back to return to the User Management Home page without creating the new user.

Note: The WLCS RDBMSrealm allows mixed case (for example: User, user) user creation.

Note: The administration tools do not allow the creation of a user with username “system” or “guest”, as these are reserved WebLogic Server terms.

Editing User Property Values

Note: Explicit properties of UUP are only editable from the administration tools if a property set is created that mirrors those properties.

To edit user property values:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Users in the Users banner. The Search for a User tool appears.

To locate a user, do one of the following:

- a. To locate the user by name, enter the username in the Username field, then click Search. The search results appear at the bottom of the page.
- b. To see a list of all users within an alphabetized category, click the appropriate letter corresponding to the first letter of the username. A list of users appear at the bottom of the page.

- c. To see a list of all users in the database, use the wildcard feature. Enter a partial username immediately followed by an asterisk (*). The asterisk is a search return variable.
3. Select the user. The User Property view appears.
4. Select a property set to view for this user. For specific instructions on Property Set Management, see [Chapter 3, “Creating and Managing Property Sets.”](#)
5. Click Search. The User Properties view appears.

The screenshot shows the 'Users: acme' interface. At the top right is a 'finished' button. Below the header, there is a section 'Select a property set to view for this user:' with a dropdown menu showing 'exampleportal' and a 'search' button. The main content area is divided into sections: 'User Information' with 'Username: acme' and an 'edit' button; 'Properties' with two items: 'FavoriteCharacter (Text, Single, Restricted)' showing 'RoadRunner' selected and 'bugs' in a text field, and 'banner_color (Text, Single, Unrestricted)' showing 'Value= #666666'. Each property has 'edit' and 'reset' buttons.

6. Click Edit on the appropriate Property bar. The associated Edit Property Values page appears.

The screenshot shows the 'Edit Property Values' page. At the top, it says 'Property Set: Property Set Name' and 'Property: Property Name'. Below is a header 'Edit Property Values' with a pencil icon. The main text says 'Select a default property value for this user and click save.' There are two radio buttons: 'True' (selected) and 'False'. At the bottom right are 'back' and 'save' buttons.

7. Change the user's values at the Edit Property Values page.
8. Click Save. A message appears indicating whether or not the edit was successful. Alternatively, click Back to return to the User Properties view without saving your changes.
9. Click Back to return to the User Properties view.
10. Return to step 4 and edit other properties as necessary.

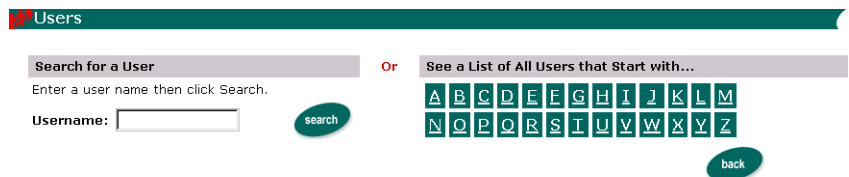
Note: If you click the Reset button on the Property bar (instead of Edit as we did in step 6), the property is set to null for that user. This will have one of three results:

- First, if the property has a default value, the user will have that default value. Note that the default value is not copied into the user's settings. The user's value is just set to null so that the default value will be returned when `getProperty()` is called for that property. If the default value changes, calling `getProperty()` will return the new default value.
- Second, if the property is defined in a Property Set but does not have a default value, the user will have a null for that property.
- Third, if the property was dynamically defined (that is, it does not belong to a Property Set), resetting causes that property to be deleted.

Deleting Users

To delete users:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Users in the Users banner. The Search for a User tool appears.



To locate a user, do one of the following:

- a. To locate the user by name, enter the username in the Username field, then click Search. The search results appear at the bottom of the page.
- b. To see a list of all users within an alphabetized category, click the appropriate letter corresponding to the first letter of the username. A list of users appear at the bottom of the page.

- c. To see a list of all users in the database, use the wildcard feature. Enter a partial username immediately followed by an asterisk (*). The asterisk is a search return variable.
3. Click the X to right of the username to delete the user. A confirmation dialog box appears.
4. Click OK to confirm the deletion.

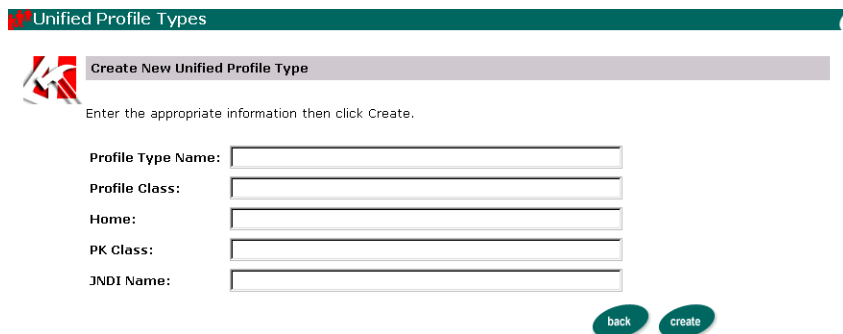
Note: When a user, declared in the `weblogic.properties` file, is deleted from the Delete Users screen, the corresponding User component and its properties will be deleted, but the username will continue to be returned from user searches.

Creating Unified Profile Types

To create unified profile types:

The Unified Profile Type tool facilitates the registration of profile types to be used as Unified User Profile (UUP) objects.

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Create in the Unified Profile Types banner. The Create New Unified Profile Type page appears.



Unified Profile Types

Create New Unified Profile Type

Enter the appropriate information then click Create.

Profile Type Name:

Profile Class:

Home:

PK Class:

JNDI Name:

back create

4 *Creating and Managing Users*

The following table contains descriptions of the Create New Unified Profile Type fields:

Field	Description
Profile Type Name	This is an arbitrary name that is used to refer to the profile type through the User Management system's <code><um:getProfile></code> JSP extension tag.
Profile Remote Interface	The remote interface of the new profile type.
Home	The home class of the new profile type.
PK Class	The primary key class of the new profile type.
JNDI Name	The JNDI lookup name of the new profile type.

3. Enter the appropriate information in the fields provided.
4. Click Create and return to the Unified Profile Types list. Alternatively, click Back to return to the User Management Home page without saving your changes.

Editing Unified Profile Types

To edit unified profile types:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click the Unified Profile Types list. The Unified Profile Type page appears.
3. Click the appropriate link to edit a unified profile type. The Edit Unified Profile Type page appears.
4. Edit the appropriate field(s) of the unified profile type.
5. Click Save and return to the Unified Profile Types list or click Back to return to the User Management Home page without saving your changes.

Deleting Unified Profile Types

To delete unified profile types:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Unified Profile Types. The Unified Profile Type page appears.
3. Click the X to right of the username to delete the user. A confirmation dialog box appears.
4. Click OK to confirm the deletion.

Using Other Realms

The remaining tools are accessible only if a realm other than Personalization Server's RDBMSRealm is used. The LDAP tools are accessible only if WebLogic's LDAPRealm is used.

Registering User Attributes for Retrieval from LDAP

This screen is used to register user attribute names for run-time retrieval via the group profile.

Note: For the LDAP features to appear in the User Management tool, you must first install and configure the WebLogic LDAP security realm for your WebLogic Server. See <http://www.weblogic.com/docs51/admindocs/ldap.html> for details.

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click LDAP in the LDAP banner. The LDAP Configuration view appears.

The screenshot shows the 'LDAP Configuration' page with a 'finished' status indicator in the top right corner. The page is divided into several sections:

- Enabled Group Attributes:** A section with a 'create' button. Below the header is a text input field for listing group attributes to be retrieved from LDAP by name.
- Enabled User Attributes:** A section with a 'create' button. Below the header is a text input field for listing user attributes to be retrieved from LDAP by name.
- LDAP Configuration Parameters - from the WebLogic LDAP Realm:** A section containing:
 - Groups Location:** A text input field with the value 'ou=Groups,o=beasys.com'. A note below states: 'Groups Location is the DN for the hierarchical parent of all relevant groups.'
 - Group Name Attribute:** A text input field with the value 'cn'. A note below states: 'Group Name Attribute is the name of the attribute which uniquely identifies groups.'

3. Click Create on the Enabled User Attributes bar. The Add User Attribute page appears.
4. Enter a new attribute to retrieve from LDAP in the User Attribute Name field.
5. Click Save. Alternatively, click Back to return to LDAP Configuration view without saving your changes.
6. Repeat steps 4 and 5 as necessary.
7. When finished, click Back.

Unregistering User Attributes for Retrieval from LDAP

To unregister user attributes for retrieval from LDAP:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click LDAP in the LDAP banner. The LDAP Configuration view appears.
3. In the Enabled User Attributes list, click the X to the right of the attribute you want to delete. A confirmation dialog box appears.
4. Click OK to confirm the deletion.
5. Repeat steps 3 and 4 as necessary.
6. When finished, click Back.

Registering Group Attributes for Retrieval from LDAP

To register group attributes for retrieval from LDAP:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click LDAP in the LDAP banner. The LDAP Configuration view appears.

The screenshot shows the LDAP Configuration interface. At the top, there is a green header bar with "LDAP Configuration" on the left and "finished" on the right. Below this, there are several sections:

- Enabled Group Attributes**: A green header bar with a "create" button on the right. Below it, the text reads: "List the group attributes that you want to be retrieved from LDAP by name."
- Enabled User Attributes**: A green header bar with a "create" button on the right. Below it, the text reads: "List the user attributes that you want to be retrieved from LDAP by name."
- LDAP Configuration Parameters - from the WebLogic LDAP Realm**: A grey header bar. Below it, there are two sub-sections:
 - Groups Location**: A grey header bar. Below it, the text reads: "Groups Location is the DN for the hierarchical parent of all relevant groups." Below this, the text reads: "Location: ou=Groups,o=beasys.com"
 - Group Name Attribute**: A grey header bar. Below it, the text reads: "Group Name Attribute is the name of the attribute which uniquely identifies groups." Below this, the text reads: "Attribute: cn"

3. Click Create on the Enabled Group Attributes bar. The Add Group Attribute tool appears.
4. Enter a new attribute in the Group Attribute Name field to retrieve from LDAP.
5. Click Save to add the attribute or click Back to return to LDAP Configuration view without saving your changes.
6. Repeat steps 4 and 5 as necessary.

Unregistering Group Attributes for Retrieval from LDAP

To unregister group attributes for retrieval from LDAP:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click LDAP in the LDAP banner. The LDAP Configuration view appears.
3. In the Enabled Group Attributes list, click the X to the right of the attribute you want to delete. A confirmation dialog box appears.
4. Click OK to confirm the deletion.
5. Repeat steps 3 and 4 as necessary.

Viewing LDAP Configuration Settings

To view LDAP configuration settings:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click LDAP in the LDAP banner. The LDAP Configuration view appears.
3. View the status of the parameters listed in the following table from the LDAP Configuration Parameters field

Parameter	Description
Groups Location	Distinguished name for the hierarchical parent of all relevant groups.
Group Name Attribute	The name of the attribute that uniquely identifies a group.
Group Username Attribute	The name of the attribute in group objects that has as its value the group members.
Users Location	Distinguished name for the hierarchical parent of all relevant users.
Username Attribute	The name of the attribute that uniquely identifies users in the system. Example: login name or unique ID.
LDAP System Principal	Distinguished name for a system level user. This user has read access to all information in the LDAP directory accessed by the application.
LDAP URL	The Universal Resource Locator (URL) of the LDAP directory server you are running.
SSL	Indicates whether communication from the Personalization Server to the LDAP directory should be encrypted over SSL.

Note: The values above are “read only” and are specified when configuring the LDAP realm.

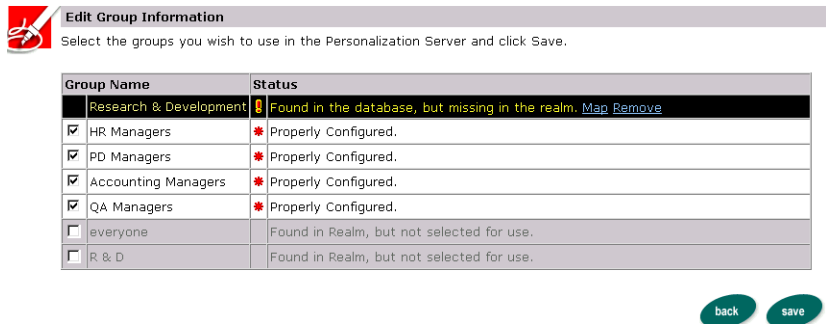
Selecting Groups for Use in the Personalization Server from the Realm

To select groups for use in the Personalization Server from the realm:

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Realm in the Realm banner. The Realm Configuration page appears.



3. Click Edit in the Groups bar. The Edit Group Information tool appears.



4. Select the group(s) you wish to use.
5. Click Save.

Mapping Realm Groups to the Personalization Server

When a name changes in the realm, you must change it in the Personalization Server too. Use this tool when a group name changes in the realm. Mapping works by changing the records in the Personalization Server to reflect the new group name.

1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Realm in the Realm banner. The Realm Configuration page appears.
3. Click Edit in the Groups bar. The Edit Group Information tool appears.
4. Click Map in the Status description of the corresponding group name. The Map Group tool appears.

Note: You are only given the option of mapping those groups that have been found in your database but are missing from the realm.

5. Select the appropriate group name from the Map To Group field.
6. Click Save. Alternatively, click Back to return to the Realm Configuration page without saving your changes.


Note: Group mapping works by simply changing the name of the group in the personalization tables to the group name in the realm. All property data is retained.

Deleting Groups from Your Database






To delete groups from your database:



1. On the Administration Tools Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Realm in the Realm banner. The Realm Configuration page appears.
3. Click Edit in the Groups bar. The Edit Group Information tool appears.

4 Creating and Managing Users

 **Edit Group Information**

Select the groups you wish to use in the Personalization Server and click Save.

Group Name	Status
<input type="checkbox"/> Research & Development	 Found in the database, but missing in the realm. Map Remove
<input checked="" type="checkbox"/> HR Managers	 Properly Configured.
<input checked="" type="checkbox"/> PD Managers	 Properly Configured.
<input checked="" type="checkbox"/> Accounting Managers	 Properly Configured.
<input checked="" type="checkbox"/> QA Managers	 Properly Configured.
<input type="checkbox"/> everyone	Found in Realm, but not selected for use.
<input type="checkbox"/> R & D	Found in Realm, but not selected for use.

4. Click Remove in the Status description of the corresponding group name.

Note: You are only given the option of deleting those groups that are found in your database but are missing from the realm. A confirmation dialog box appears.

5. Click OK to confirm the deletion.

Deleting User Records That Do Not Exist in the Realm from the Personalization Database

To delete user records that do not exist in the realm from the Personalization database:

1. On the Administration Tool Home page, click the User Management icon. The User Management Home page appears.
2. On the User Management Home page, click Realm in the Realm banner. The LDAP Configuration view appears.
3. Click Edit in the Users bar. The Clean Up Users tools appears with a count of users found in the personalization database but not in the realm.
4. Click Clean Up if the usernames are no longer needed. All associated records are removed.

5 Creating and Managing Content

The Content Manager provides content and document management capabilities for use in personalization services. The Content Manager works with files or with content managed by third-party vendor tools from Documentum and Interwoven.

This topic includes the following sections:

- What Is the Content Manager?
 - Using Third-party Tools
 - Constructing Queries Using Java
 - Differences Between Content Management and Document Management
 - Using the Document Servlet
 - JSP Tags
- Configuring the Content Manager
 - Configuring the Document Schema EJB Deployment Descriptor
 - Configuring the DocumentManager EJB Deployment Descriptor
 - Setting Up Connection Pools
 - Configuring WebLogic Commerce Properties
 - Using the Show Document Servlet
 - Querying Document Content
 - Structuring a Query
 - Using Comparison Operators to Construct Queries

- Using the BulkLoader to Load File-based Content
- Using Content Management JSP Tags

What Is the Content Manager?

The Content Manager runtime subsystem provides access to content via both tags and EJBs. The Content Management tags allow a JSP developer to receive an enumeration of Content objects by querying the content database directly using a search expression syntax.

The Content Manager component works alongside the other components to deliver personalized content, but does not have a GUI-based tool for edit-time customization. The content engine behind the `ContentManager` may be set up to be the reference implementation, provided out of the box, or Documentum. The Content Management component supports querying that returns content from a content repository using several methods:

- **Search for content by metadata**—boolean logic searching evaluates content that matches a metadata/operator/value criteria.
- **Retrieve content by ID**—the system allows retrieval of raw bytes of content data—either in blocks or in its entirety—through the content’s known identifier.
- **Query content metadata by ID**—the system, through the known identifier of a content piece, can query the metadata describing the content piece. Several metadata attributes provide information about the content. The query language maps some attribute names onto explicit attributes of the `Content` or `Document` objects the query searches. Queries searching for `Content` objects support the following case-sensitive explicit attribute names:
 - *identifier*: Corresponds to the unique `String` identifier of the `Content` (that is, the `getIdentifier` method).
 - *mimeType*: Corresponds to the `String` MIME type of the `Content` (that is, the `getMimeType` method).
- Queries searching for `Document` objects support the following additional case-sensitive explicit attribute names:

- *size*: Corresponds to the `Long` size of the document in bytes (that is, the `getSize` method). Documents without file bytes will have a size of 0 or less.
- *version*: Corresponds to the `Integer` version number of the document (that is, the `getVersion` method).
- *author*: Corresponds to the `String` identifier of the author of the document (that is, the `getAuthor` method).
- *creationDate*: Corresponds to the `Timestamp` of when the document was created (that is, the `getTimestamp` method).
- *modifiedBy*: Corresponds to the `String` identifier of the individual who last modified the document (that is, the `getModifiedBy` method).
- *modifiedDate*: Corresponds to the `Timestamp` of when the document was last modified (that is, the `getModifiedDate` method).
- *lockedBy*: Corresponds to the `String` identifier of the individual who has the document locked (that is, the `getLockedBy` method).
- *description*: Corresponds to the `String` description of the document (that is, the `getDescription` method).
- *comments*: Corresponds to any `String` comments about the document (that is, the `getComments` method).

Note: All other attribute names in queries are considered implicit metadata properties.

- **Get content schema by name**—the document management system (DMS) contains a set of named schemas that describe a set of non-standard metadata attributes. Each piece of content in the DMS is associated with one of these schemas and each schema specifies valid attributes
- **Get content schema names**—a user can query the system for a list of all schema names a DMS supports.

Note: See “Querying Document Content” on page 5-14 for more information about queries.

Using Third-party Tools

BEA partners with third-party vendors to add flexibility to the WebLogic Personalization Server. The Content Manager works with Interwoven's Teamsite/OpenDeploy product and Documentum's 4i product. Both these products provide robust, content-creation management solutions while the Content Manager personalizes and serves the content to the end user.

How Do I Choose What Content Management Tools to Use?

- **WebLogic Personalization Server BulkLoader**—for sites with limited content personalization needs and existing metatagged HTML, WebLogic Personalization Server includes a command-line utility called the BulkLoader. The BulkLoader can parse a directory of HTML files and store their URL address and metadata attributes in a JDBC store. The BulkLoader automatically creates the schema for these attributes.

Interwoven Teamsite/OpenDeploy Integration—for customers who have larger amounts of content and want more control over the publishing and tagging of content, WebLogic Personalization Server provides integration with the Interwoven Teamsite/OpenDeploy product. Teamsite is a content capture, versioning, staging, and publishing system. Teamsite templates define the metadata attributes for the content; customers can use the templates to categorize the documents during the creation process. Teamsite is not a run-time engine; it does not actually get queried by the WebLogic Personalization Server. Once content is captured, the Interwoven OpenDeploy workflow capability works with the WebLogic Personalization Server to publish content to a well-known location and the metadata to our JDBC store.

A WebLogic Personalization Server customer can purchase the Interwoven Teamsite product for content management and publishing. The extensions for Interwoven are available from the BEA [download center](#) as one of the Commerce Servers download options.

- **Documentum 4i Integration**—as a compatible document management solution, BEA provides integration with the Documentum 4i product. Documentum products manage the metadata and documents in their own repositories.

Content developers working with Documentum create metadata capture tools in Documentum 4i. When they complete the authoring process and check in their documents to Documentum, they must tag the content. Depending on

administrative options, this content can be available immediately to the WebLogic Personalization Server for display on the customer's e-commerce site.

WebLogic Personalization Server customers can purchase Documentum 4i to manage the documents or content for their e-commerce site. The JDBC driver for Documentum 4i is available from the BEA [download center](#) as one of the Commerce Servers download options.

Constructing Queries Using Java

To construct queries using Java syntax instead of using the query language supplied with the Content Management component, refer to the [Javadoc API documentation](#).

Note: Use the constants in `TypesHelper` when calling `Logical.setLogical` and `Criteria.setComparator`.

The `ContentManager` session bean is the primary interface to the functionality of the Content Management component. Using a `ContentManager` instance, content is returned based on a `Search` object with an embedded `Expression`. An `Expression` is a Boolean tree of arbitrary depth, with other sub-`Expressions` as nodes. The `Expression` interface is meant to be abstract, where the actual instances are `Logical` or `Criteria` interfaces. As an example, the expression `color == 'red' && price > 50` would consist of a `Logical` with the value *and* that has as children two `Criteria`.

Differences Between Content Management and Document Management

`Content` objects include metadata about the content. Metadata provides a means to query and match content with users by allowing the system to retrieve content based on the metadata that describes the content. In general, some kind of content management system provides services such as retrieval of content and content authoring services including creation, editing, versioning, and workflow.

`Documents` are a specialized type of `Content` that provide two methods for retrieval: a metadata-searching mechanism and retrieval of the pure bytes of the document's file. `Documents` should include additional explicit metadata properties related to the file

and its versioning, including its size, name, path, author, and version. A document management system usually provides document-based services for documents that reside in the system's repository.

WebLogic Personalization Server provides the entire `Content` object model; however, it only provides the `Document` object as a concrete implementation (subclass) of the `Content` class.

Using the Document Servlet

The Content Management component includes a servlet capable of outputting the contents of a `Document` object. This servlet is useful when streaming the contents of an image that resides in a content management system or to stream a document's contents that are stored in a content management system when an HTML link is selected. The servlet supports the following Request/URL parameters:

Request Parameter	Required	Description
<code>contentHome</code>	Maybe	If the <code>contentHome</code> initialization parameter is not specified, then this is required and will be used as the JNDI name of the <code>DocumentHome</code> . If the <code>contentHome</code> initialization parameter is specified, this is ignored.
<code>contentId</code>	No	The string identifier of the <code>Document</code> to retrieve. If not specified, the servlet looks in the <code>PATH_INFO</code> .
<code>blockSize</code>	No	The size of the data blocks to read. The default is 8K. Use 0 or less to read the entire block of bytes in one operation.

The servlet only supports `Documents`, not other subclasses of `Content`. It sets the `Content-Type` to the `Document`'s `mimeType` and, the `Content-Length` to the `Document`'s size, and correctly sets the `Content-Disposition`, which should present the correct filename when the file is saved from a browser.

Example 1: Usage in a JSP

This example searches for news items that are to be shown in the evening, and displays them in a bulleted list.

```
<cm:select
contentHome="<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME%" max="5"
sortBy="creationDate ASC, title ASC"
query="type = 'News' && timeOfDay = 'Evening' && mimeType like
'text/*' " id="newsList" />

<ul>
  <es:forEachInArray array="<%=newsList%" id="newsItem"
type="com.beasys.commerce.axiom.content.Content">
    <li><a href="/showDocServlet/<cm:printProperty
id="newsItem" name="identifier" encode="url"/>
      &contentHome=<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME%">
      <cm:printProperty id="newsItem" name="title"
      encode="html"/></a>
    </es:forEachInArray>
</ul>
```

Example 2: Usage in a JSP

This example searches for image files that match keywords that contain *bird* and displays the image in a bulleted list.

```
<cm:select
contentHome="<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME%">"
max="5" sortBy="name" id="list" query="Keywords like '*birds*' &&
mimeType like 'image/*' " />
<ul>
  <es:forEachInArray array="<%=list%" id="img"
type="com.beasys.commerce.axiom.content.Content">
    <li>
      &contentHome=<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME%">">
    </es:forEachInArray>
</ul>
```

JSP Tags

The Content Management component includes the following four JSP tags. These tags allow a JSP developer to include non-personalized content in a HTML-based page. Note that none of the tags support or use a body.

- The `<cm:select>` tag uses only the search expression query syntax to select content. See the “[JSP Tag Library Reference](#)” in the *WebLogic Personalization Server Developer’s Guide* for more information.
- The `<cm:selectById>` tag retrieves content using the content’s unique identifier. See the “[JSP Tag Library Reference](#)” in the *WebLogic Personalization Server Developer’s Guide* for more information.
- The `<cm:printProperty>` tag inlines the value of the specified Content metadata property as a string. See the “[JSP Tag Library Reference](#)” in the *WebLogic Personalization Server Developer’s Guide* for more information.
- The `<cm:printDoc>` tag inlines the raw bytes of a Document object into the JSP output stream. See the “[JSP Tag Library Reference](#)” in the *WebLogic Personalization Server Developer’s Guide* for more information.

Configuring the Content Manager

The DocumentSchema EJB and DocumentManager EJB deployment descriptors handle the configuration for the Content Management component. To use the reference implementation document repository, you need to configure the EJB deployment descriptors and also set up two WebLogic Server JDBC connection pools.

Once the deployment descriptor has been written, just build the EJBs as you normally would, then add the resulting JAR file to your `ejb.deploy` entry in the `weblogic.properties` file.

Configuring the Document Schema EJB Deployment Descriptor

The logic for loading Document Schema EJBs is handled via a `SmartBMP`. The Schema EJB implementation loads the `SmartBMP` object from a class name specified in the EJB environment in the EJB's deployment descriptor. The EJB environment variable is `SmartBMPClass`. The value must be the fully qualified class name of the `SmartBMP` to use. This `SmartBMP` must be capable of populating a `SchemaImpl` object with `PropertyMetaData` objects.

To use the reference implementation document management system, set `SmartBMPClass` to

`com.beasys.commerce.axiom.document.SPISchemaSmartBMP` and specify the following EJB environment variables in the document EJB deployment descriptor:

- `SmartBMPUpdate`: Set to false. (no italics!)
- `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 (see the topic [“Setting Up Connection Pools” on page 5-11](#)). This should correspond to the Data Source attached to the WebLogic connection pool that uses the document reference implementation JDBC driver.
- `jdbc/commercePool`: A `DataSource` reference to the `weblogic.jdbc.jts.commercePool`, which should be attached to the WebLogic connection pool `commercePool`.

Other `SmartBMP` classes for other document management systems will possibly require more and/or different EJB environment variables.

Configuring the DocumentManager EJB Deployment Descriptor

The `DocumentManagerSession` EJB simply hides the details of getting to the `Document` and `DocumentSchema` EJBs. It understands the following environment variables in its deployment descriptor:

- *PropertyCase*: This sets how the `DocumentImpl` modifies incoming property names. If this is *lower*, all property names are converted to lower case. If this is *upper*, all property names are converted to upper case. 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 lower case). For the document reference implementation, do not specify the *PropertyCase*.
- *jdbc/docPool*: A Data Source reference to the document JDBC connection Pool (see the topic “[Setting Up Connection Pools](#)” on page 5-11). This should correspond to the Data Source attached to the WebLogic connection pool that uses the document reference implementation JDBC driver.
- *ejb/ContentHome*: EJB reference to the Document Home to which this should delegate for non-readOnly access.

Note: Since the Document EJB is deprecated for read access, this will eventually no longer be required.

- *ejb/SchemaHome*: EJB reference to the Schema Home to which this should delegate for Schema information.
- *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`.

Setting Up Connection Pools

For the document reference implementation, set up a specialized WebLogic connection pool and DataSource which will be used by the DocumentManager via the `jdbc/docPool` reference. (See the topic [“Configuring the DocumentManager EJB Deployment Descriptor”](#) on page 5-10.)

For example, if the connection pool name is `docPool`:

- The *URL* should be
`jdbc:beasys:docmgmt:com.beasys.commerce.axiom.document.ref.RefDocumentProvider.`
- The *driver* should be
`com.beasys.commerce.axiom.document.jdbc.Driver`. It should not be configured to use a `test_table`, although it can be allowed to shrink. The driver supports the following properties:
 - *jdbc.url*: (Required) Specifies the JDBC URL of the database. The connection in this pool opens a connection to this JDBC URL. This property probably should refer to another, non-specialized JDBC connection pool, although it can be any JDBC URL.
 - *jdbc.driver*: Specifies a JDBC driver class name to load.
 - *jdbc.isPooled*: If `true`, then the system assumes the JDBC URL in *jdbc.url* is a pooling connection URL and connections will open and close as needed. If `false`, then this connection opens one connection via the *jdbc.url* and uses that for its lifetime. If the *jdbc.url* starts with `jdbc:weblogic:pool` or `jdbc:weblogic:jts`, then this property automatically becomes `true`.
 - *docBase*: (Required) Specifies the document base of the document files. The IDs in the database use file paths relative to this directory and must exist when the connection is created. To operate in a cluster or a multi-server environment, you must either replicate the files on the machines or put the *docBase* directory on a shared volume.
 - *schemaXML*: Specifies the file or directory where the XML schema (following the `doc-schemas.dtd`) resides. Either the *schemaXML* property or the *iw.schemaBase* property is required, although the schemas under *schemaXML* take precedence if both are specified. The *schemaXML* property has the same constraints as the *docBase* property when used in a cluster.

Note: If *schemaXML* is a directory, the connection will recurse under it and load all files ending in *.xml* (* .xml).

Note: If *schemaXML* is a file, the connection loads it.

- *iw.schemaBase*: Specifies the directory in which the InterWoven *datacapture.cfg* files reside. The connection recurses through this directory, loading all *datacapture.cfg* files it finds. Either the *iw.schemaBase* or *schemaXML* property is required, although you can specify both. The *iw.schemaBase* property has the same constraints as the *docBase* property when used in a cluster.
- Set up a non-transactional *DataSource* pointing to the pool. The name of the *DataSource* should be the same as that configured with the *DocumentManager* and *Schema*.

All other properties are passed with *jdbc.url* when the *Driver Manager* opens a database connection.

Example connection pool Entry

The following example shows a sample configuration in the *weblogic.properties* file.

```
weblogic.jdbc.connectionPool.docPool=\
url=jdbc:beasys:docmgmt:com.beasys.commerce.axiom.document.ref.RefDocumentProvider,\
driver=com.beasys.commerce.axiom.document.jdbc.Driver,\
loginDelaySecs=1,\
initialCapacity=1,\
maxCapacity=5,\
capacityIncrement=1,\
allowShrinking=true,\
shrinkPeriodMins=15,\
refreshMinutes=10,\
    props=jdbc.url=jdbc:weblogic:pool:commercePool;\
    jdbc.isPooled=true;\
    docBase=C:/WeblogicCommerce/docBase;\
    schemaXML=C:/WeblogicCommerce/docSchemas;\
    iw.schemaBase=C:/iw-home/templatedata
weblogic.allow.reserve.weblogic.jdbc.connectionPool.docPool=every
one
weblogic.jdbc.DataSource.weblogic.jdbc.pool.docPool=docPool
```


Configuring WebLogic Commerce Properties

Use a `ContentManager` or `DocumentManager` with `<cm:select>` or `<cm:selectById>` to retrieve Content or Documents. The default `DocumentManager` is deployed at `com.beasys.commerce.axiom.document.DocumentManager`.

To help with the JNDI names, the `ContentHelper` class has the following six constants:

`DEF_CONTENT_HOME`

Specifies the default deployed `ContentHome`.

`DEF_CONTENT_MANAGER_HOME`

Specifies the default deployed `ContentManagerHome`.

`DEF_CONTENT_SCHEMA_HOME`

Specifies the default deployed `SchemaHome` for Content.

`DEF_DOCUMENT_HOME`

Specifies the default deployed `DocumentHome`.

`DEF_DOCUMENT_MANAGER_HOME`

Specifies the default deployed `DocumentManagerHome`.

`DEF_DOCUMENT_SCHEMA_HOME`

Specifies the default deployed `SchemaHome` for Document.

The values of those constants are read from the `weblogiccommerce.properties` file from the values for the following properties:

`DEF_CONTENT_HOME`

`commerce.home.content.ContentHome`

`DEF_CONTENT_MANAGER_HOME`

`commerce.home.content.ContentManagerHome`

`DEF_CONTENT_SCHEMA_HOME`

`commerce.home.content.ContentSchemaHome`

`DEF_DOCUMENT_HOME`

`commerce.home.document.DocumentHome`

`DEF_DOCUMENT_MANAGER_HOME`

`commerce.home.document.DocumentManagerHome`

`DEF_DOCUMENT_SCHEMA_HOME`

`commerce.home.document.DocumentSchemaHome`

Therefore, in any `<cm:select>`, `<cm:selectById>`, `<pz:contentQuery>` or `<pz:contentSelector>` tags, define the `contentHome` (or `contenthome`) parameter to use a `ContentManagerHome` or `DocumentManagerHome`.

Example:

The News Index and News Viewer portlets use the default deployed `DocumentManager` and can be used as a reference. The JSPs are located in the `server/public_html/portals/repository/portlets` directory in the `news_index.jsp`, `news_viewer.jsp` and `content_titlebar.jsp` files.

Using the Show Document Servlet

To operate the Show Document servlet, it should be registered with WebLogic Server. The class name of the servlet is

`com.beasys.commerce.content.ShowDocServlet`. To register it with WebLogic, add a line similar to the following to your `weblogic.properties` files:

```
weblogic.httpd.register.showDocServlet=\
    com.beasys.commerce.content.ShowDocServlet
```

Reference the class in the URL as `/showDocServlet`. To change the URL reference, change `/showDocServlet`. For example, to specify the URL as

`/myapp/doc-shower`, enter the following in the `weblogic.properties` file:

```
weblogic.httpd.register.myapp/doc-shower=\
    com.beasys.commerce.content.ShowDocServlet
```

Querying Document Content

There are several way to query the document management system. To query the system, you construct a query expression, then pass the expression to any one of these:

- JSP tags (see [“Using Content Management JSP Tags”](#) on page 5-25.)
- `ContentHelper` (see the [Javadoc API documentation](#))
- `ContentManager` (see the [Javadoc API documentation](#))
- `ContentHome` (see the [Javadoc API documentation](#))

Structuring a Query

WebLogic Personalization Server queries use a syntax similar to the SQL string syntax that supports basic Boolean-type comparison expressions, including nested parenthetical queries. In general, the template for use includes a metadata property name, a comparison operator, and a literal value. The basic query uses the following template:

```
attribute_name comparison_operator literal_value
```

Note: Consult the [Javadoc API documentation](#) on `com.beasys.commerce.util.ExpressionHelper` for more information about the query syntax.

Several constraints apply to queries constructed using this syntax:

- String literals must be enclosed in single quotes.
 - `'WebLogic Server'`
 - `'football'`
- Date literals can be created via a simplistic `toDate` method that takes one or two `String` arguments (enclosed in single quotes). The first, if two arguments are supplied, is the `SimpleDateFormat` format string; the second argument is the date string. If only one argument is supplied, it should include the date string in `'MM/dd/yyyy HH:mm:ss z'` format.
 - `toDate('EE dd MMM yyyy HH:mm:ss z', 'Thr 06 Apr 2000 16:56:00 MDT')`
 - `toDate('02/23/2000 13:57:43 MST')`
- Use the `toProperty` method to compare properties whose names include spaces or other special characters. In general, use `toProperty` when the property name does not comply with the Java variable-naming convention that uses alphanumeric characters.
 - `toProperty('My Property') = 'Content'`
- To include a scope into the property name, use either `scope.propertyName` or the `toProperty` method with two arguments.
 - `toProperty('myScope', 'myProperty')`

Note: The reference document management system ignores property scopes.

- Use `\` along with the appropriate character(s) to create an escape sequence that includes special characters in string literals.
 - `toProperty ('My Property\'s Contents') = 'Content'`
 - Additionally, use Java-style unicode escape sequences to embed non-ASCII characters in string literals.
 - Description like `'*\u65e5\u672c\u8a9e*'`
- Note:** The query syntax can only contain ASCII and extended ASCII characters (0-255).
- Note:** Use `ExpressionHelper.toStringLiteral` to convert an arbitrary string to a fully quoted and escaped string literal which can be put in a query.
- The `now` keyword—only used on the literal value side of the expression—refers to the current date and time.
 - Boolean literals are either `true` or `false`.
 - Numeric literals consist of the numbers themselves without any text decoration (like quotation marks). The system supports scientific notation in the forms (for example, `1.24e4` and `1.24E-4`).
 - An exclamation mark (!) can be placed at an opening parenthesis to negate an expression.
 - `!(keywords contains 'football') || (size >= 256)`
 - The Boolean and operator is represented by the literal `&&`.
 - `author == 'james' && age < 55`
 - The Boolean or operator is represented by the literal `||`.
 - `creationDate > now || expireDate < now`

The following examples illustrate full expressions:

Example 1:

```
((color='red' && size <=1024) || (keywords contains 'red' && creationDate < now))
```

Example 2:

```
creationDate > toDate ('MM/dd/yyyy HH:mm:ss', '2/22/2000 14:51:00')  
&& expireDate <= now && mimetype like 'text/*'
```

Using Comparison Operators to Construct Queries

To support advanced searching, the system allows construction of nested Boolean queries incorporating comparison operators. The following table summarizes the comparison operators available for each metadata type. (For more information about the native types supported in WebLogic Personalization Server, see [Support for Native Types](#) in the *Developer's Guide* chapter [Overview of Personalization Development](#) .)

Operator Type	Characteristics
Boolean (==, !=)	Boolean attributes support an equality check against boolean.TRUE or Boolean.FALSE. (
Numeric (==, !=, >, <, >=, <=)	Numeric attributes support the standard equality, greater than, and less than checks against a <code>java.lang.Number</code> .
Text (==, !=, >, <, >=, <=, like)	Text strings support standard equality checking (case sensitive), plus lexicographical comparison (less than or greater than). In addition, strings can be compared using wildcard pattern matching (that is, the <i>like</i> operator), similar to the SQL LIKE operator or DOS prompt file matching. In this situation, the wildcards will be * (asterisk) for match any and ? (question mark) for match single. Interval matching (for example, using []) is not supported. To match * or ? exactly, the quote character will be \ (backslash).
Datetime (==, !=, >, <, >=, <=)	Date/time attributes support standard equality, greater than, and less than checks against a <code>java.sql.Timestamp</code> .
Multi-valued Comparison Operators (contains, containsall)	<p>Multi-valued attributes support a <i>contains</i> operator that takes an object of the attribute's subtype and checks that the attribute's value contains it. Additionally, multi-valued attributes support a <i>containsall</i> operator, which takes another collection of objects of the attribute's subtype and checks that the attribute's value contains all of them.</p> <p>Single-valued operators applied to a multi-valued attribute should cause the operator to be applied over the attribute's collection of values. Any value that matches the operator and operand should return <code>true</code>. For example, if the multi-valued text attribute <i>keywords</i> has the values <i>BEA</i>, <i>Computer</i>, and <i>WebLogic</i> and the operand is <i>BEA</i>, then the <code><</code> operator returns <code>true</code> (<i>BEA</i> is less than <i>Computer</i>), the <code>></code> operator returns <code>false</code> (<i>BEA</i> is not greater than any of the values), and the <code>==</code> operator returns <code>true</code> (<i>BEA</i> is equal to <i>BEA</i>).</p>

Operator Type	Characteristics
User Defined Comparison Operators	Currently, no operators can be applied to a user-defined attribute.

Note: The search parameters and expression objects support negation of expressions via a bit flag (!).

Note: The reference document management system has only single-value Text and Number properties. All implicit properties are single-value Text.

Using the BulkLoader to Load File-based Content

WebLogic Personalization Server provides no run-time tools to load metadata information from a content database. However, the server provides a command-line utility, the BulkLoader, that descends a directory hierarchy, parses the HTML-style <meta> tags, reverses the metadata content contained within the <meta> tags into schema information, and loads the resulting documents into the reference implementation database.

The BulkLoader is a command-line application that is capable of loading document metadata into the reference implementation database from a directory and file structure. The BulkLoader parses the document base and `weblogic.properties` and loads all the document metadata so that the Content Management component can search for documents. The BulkLoader supports all document types, not just HTML documents.

Command Line Usage

The BulkLoader class allows a number of command-line switches:

```
java com.beasys.commerce.axiom.document.loader.BulkLoader
[-/+verbose] [-/+recurse] [-/+delete] [-/+metaparse] [-/+cleanup]
[-/+hidden] [-/+inheritProps] [-schemaName <name>] [-encoding <encoding>]
[-properties <name>] -conPool <name> [-schema <name>] [+schema]
[-match <pattern>] [-ignore <pattern>] [-htmlPat <pattern>]
[-d <dir>] [-mdext <ext>] [--]
[files... directories...] [-filter <filter class>] [+filters]
```

`-verbose`

Emits verbose messages.

- `+verbose`
Runs quietly [default].
- `-recurse`
Recurse into directories [default].
- `+recurse`
Does not recurse into directories.
- `-delete`
Removes document from database.
- `+delete`
Inserts documents into database [default].
- `-metaparse`
Parses HTML files for `<meta>` tags [default].
- `+metaparse`
Does not parse HTML files for `<meta>` tags.
- `-cleanup`
If specified, this only performs a table cleanup using the `-d` argument as the document base. (All files will need to be under that directory.)
- `+cleanup`
Turns off table cleanup (do a document load) [default].
- `-hidden`
Specifies to ignore hidden files and directories [default].
- `+hidden`
Specifies to include hidden files and directories.
- `-inheritProps`
Specifies to have metadata properties be inherited when recursing [default].
- `+inheritProps`
Specifies to have metadata properties not be inherited when recursing.
- `-htmlPat <pattern>`
Specifies a pattern for determining which files are HTML files when determining whether to do the `<meta>` tag parse. This can be specified multiple times. If none are specified, `*.htm` and `*.html` are used.
- `-properties <name>`
Specifies the location of the `weblogic.properties` file which should contain the `connectionPool` definition. Defaults to `weblogic.properties` in the current directory.

- `-conPool <name>`
Specifies the `connectionPool` name from the properties file from which the BulkLoader should get the connection information.
- `-schema <name>`
Specifies the path to the schema file the BulkLoader will generate (defaults to `document-schema.xml`).
- `+schema`
If specified, then no schema file will be created.
- `-schemaName <name>`
Specifies the name of the schema generated by the BulkLoader. Defaults to "LoadedData".
- `-encoding <name>`
Specifies the file encoding to use. Defaults to your system's default encoding. (See your JDK documentation for the valid encoding names.)
- `-match <pattern>`
Specifies a file pattern the BulkLoader should include. This can be specified multiple times. If none are specified, all files and directories are included.
- `-ignore <pattern>`
Specifies a file pattern the BulkLoader should not include. This can be specified multiple times.
- `-d <dir>`
Specifies the `docBase` that non-absolute paths will be relative to. If not specified, "." (current directory) is used.
- `-mdext <ext>`
Specifies the file name extension for metadata property files. The value should start with a "." (defaults to `.md.properties`).
- `-filter <filter class>`
Specifies the class name of a `LoaderFilter` to run files through. This can be specified multiple times to add to the list of Loader Filters.
- `+filters`
Clears the current list of Loader Filters. (This will clear the default filters as well.)
- `--`
Everything after this is considered a file or directory.

How the BulkLoader Finds Files

The following sequence describes how the BulkLoader locates files:

1. The BulkLoader starts by looking at the list of files and directories specified from the command line.

- If no files or directory are specified, it uses only the `docBase` specified by the `-d` option. It then loops over the list of files and directories.
- If it finds a directory and `+recurse` is specified, then it stops.
- If it finds a directory and recursion is turned on (the default or with `-recurse`), then the BulkLoader loops over the files and directories contained within that directory.

Note: If the file or directory is not an absolute path, then it is assumed to be relative to the `docBase` specified by the `-d` option.

2. To determine if the BulkLoader should process a file or directory, it checks to see if the file is marked as a hidden file.

Note: If it is a hidden file (or directory) and the `+hidden` option was not specified, then the file or directory is ignored.

3. If the file or directory does not exist or is not readable by the user executing the BulkLoader, a warning is displayed and the file or directory is ignored.

4. If the file or directory is a file, then it is loaded.

5. If the loaded object is a directory and recursion is enabled, then the files and directories under the directory are retrieved by filtering against the `-match` and `-ignore` options.

Note: The `-match` and `-ignore` options only apply to files and directories not listed on the command line; in other words, they apply only to those found by recursing into a directory. The patterns specified with the `-match` and `-ignore` options (and the `-htmlPat` options, for that matter) should be DOS-style patterns: `*` matches any set of characters, `?` matches any one character. Sets of characters (for example, `[aceg]`) are not supported.

6. If the subfile or directory name matches any of the patterns specified by a `-ignore` option, the subfile or directory is ignored.

7. If the subfile or directory is a directory, then it is included.

8. If the subfile or directory is a file and no `-match` options were specified, then it will be included; if at least one `-match` option is supplied, then the filename must match at least one of `-match` patterns.

Note: Files with an extension matching the extension specified by `-mdext` (*.md.properties* by default) are always ignored.

How the BulkLoader Finds Metadata Properties

As the BulkLoader is finding files and directories, it will also attempt to load metadata property files. Whenever the BulkLoader encounters a directory that it will process, it looks for a file called `dir.<mdext>` where `<mdext>` is the extension specified by the `-mdext` option. Therefore, the default filename it looks for is `dir.md.properties`. If this file exists and is readable by the user, the BulkLoader loads it as a Java-style properties file of `name=value` properties. If the directory is actually a subdirectory entered because `+recurse` was not specified and the `+inheritProps` option is not specified, then the properties from `dir.md.properties` will be added to the properties from the parent directories. All files in the directory gain these metadata properties.

When the BulkLoader finds a file which is to be included and loaded, it looks for a file whose name is the original file name appended with the `-mdext` extension. So, by default, if the file is called `image.gif`, the BulkLoader looks for a file called `image.gif.md.properties`. If that file exists and is readable, the BulkLoader loads those properties into the directory's properties (and possibly the parent directories' as well).

Next, if the file is an HTML file and the `+metaparse` option was not specified, then the BulkLoader will parse the HTML, looking for `<meta>` tags and `<title>` tags. The BulkLoader determines if a file is an HTML file by using the filename patterns specified by the `-htmlPat` options. If no `-htmlPat` patterns are specified, then `*.htm` and `*.html` are used. The BulkLoader will load into the file's properties any `<meta>` tags that contain name and content values found anywhere in the file (not just in the HTML head section). Additionally, it will pull the title from the `<title></title>` and set it as `"title"`.

Finally, the BulkLoader will pass the file to the `loadProperties` method of each registered `LoaderFilter` (the `-filter` option). The `LoaderFilter` may assign additional metadata to the file. When the BulkLoader starts up, it looks for a `com/beasys/commerce/axiom/document/loader/loader.properties` file in the classpath. From that, it looks for a `loader.defFilters` property. This is the colon-separated list of `LoaderFilter` class names the BulkLoader should always

load. Unless that file is modified, the BulkLoader will load an `ImageLoaderFilter`, which will pull the width and height from `*.gif`, `*.jpg`, `*.png`, and `*.xbm` image files.

In summary, the BulkLoader gathers metadata for a document from the following sources (in this order):

1. The parent directories `dir.md.properties` file.
2. The file's directory's `dir.md.properties` file.
3. The file's `.md.properties` file.
4. If the file is an HTML file, then it uses `<meta>` tags.
5. The list of `LoaderFilters`.

From there, the ID of the document in the database will be the file path, relative to the `docBase` specified by the `-d` option. If the file path is not relative to the `docBase`, then it will be relative to the path from the command line. The file size will be retrieved from the file. The `mimeType` will be determined by the file's extension. The `modifiedDate` in the database will become the current time (since that is when the document is being modified in the database).

Cleaning Up the Database

If the `-cleanup` option is specified, the BulkLoader will not actually load any documents. Instead, it will attempt to clean up and update the database tables. It will first query the database, looking for any metadata entries that do not have corresponding document entries. For each of those, it will create a document entry. It will then go over each document entry and update the size, modified date, and possibly the MIME type (if the MIME type is not in the database) based upon the files in the `docBase` specified with the `-d` option.

Loading Internationalized Documents

The BulkLoader accepts a `-encoding <enc>` option. When this is specified, the BulkLoader will use that encoding to open all HTML files to find `<meta>` tags.

For example, if the files under the `Unicode-files` directory were saved in the Unicode encoding, you could do:

```
java com.beasys.commerce.axiom.document.loader.BulkLoader -verbose
-properties weblogic.properties -conPool commercePool -schema
```

`dmsBase\schemas\unicode-files.xml -d dmsBase unicode-files -encoding Unicode`. When `-encoding` is specified, the generated schema XML file will be in the UTF-8 encoding (since some metadata property names might not be ASCII), which the run-time engine can read in. (Note: UTF-8 is a superset of ASCII and can be mostly read by common text editors.)

When `-encoding` is specified, all HTML files the BulkLoader encounters will be opened with the specified encoding. Therefore, either the encoding must be a superset of all the files' encodings (for example, ISO8859_1 is a superset of ASCII, where as Unicode is not) or the BulkLoader might not be able to correctly pull out the META tag information. It is recommended to either save all documents in a single encoding or to run the BulkLoader against only certain directories at a time (for example, put all the Big5 files in one directory).

The list of available encoding names is contained in the documentation for your JDK, or the documentation for the tool which created the file. If you are not creating files containing non-ASCII characters, this should not affect you. If you want to check if the BulkLoader is correctly parsing your HTML file, you can use the

`com.beasys.commerce.axiom.document.loader.MetaParser` class. For example:

```
java com.beasys.commerce.axiom.document.loader.MetaParser
unicode.htm unicode would print out the <meta> tags found in the unicode.htm
file, assumed to be Unicode encoded. Of course, any non-ASCII character probably
will not print correctly to your console window, but you can tell what it thinks it found.
```

Generating Schema Files

Additionally, the BulkLoader supports a `-schemaName <name>` argument which controls the name of the schema in the generated XML file; this in turn affects the name of the Content PropertySets which appear in the rules editor. If not specified, it defaults to "LoadedData."

After loading all the documents on the list, if the `+schema` option is not specified, the BulkLoader will output a XML file containing the schema information and following the doc-schemas DTD. The BulkLoader will output a single schema which contains entries for all the metadata attributes it finds over the entire load.

If `+schema` is specified, then no schema file will be created.

Using Content Management JSP Tags

To use the Content Management JSP tags, ensure that the `cm.tld` file resides in the `WEB-INF` directory of your WAR files or in your document root.

Content Cache

The `<cm:select>` and `<cm:selectById>` tags support a session-based, per-user Content cache for content searches. To enable this, both tags support the following parameters. All three tag parameters can be JSP run-time expressions.

`useCache`

Set to `true` or `false`. The default is `false`. If `true`, then the `ContentCache` will be used (it will be stored in the user's `HttpSession`).

`cacheId`

The ID name to use to cache the Content under. Internally, the cache is implemented as a `Map`; this will become the key. If this is not specified, than the `id` parameter of the tag will be used.

`cacheTimeout`

The time, in milliseconds, for which the cached Content is valid. If more than this amount of time has passed since the Content was cached, the cached Content will be cleared, retrieved, and placed back in the cache. Use `-1` for no timeout (always use the cached Content); `0` to immediately timeout the cached Content.

Additionally, the `commerce.content.cache.useSoftHashMap` property in the `weblogiccommerce.properties` controls whether the `ContentCaches` internally use `SoftReferences` to cache the Content. `SoftReferences` should allow the garbage collector to clear portions of the caches as memory is needed. This defaults to `false`.

Note: The Windows NT 1.2.2 JVM supports `SoftReference` quite well; however, the Solaris Production 1.2.1_04 JVM always immediately clears `SoftReferences`, thereby eliminating their usefulness as a caching mechanism. It is recommended that `commerce.content.cache.useSoftHashMap` be set to `true` under Windows NT, but set to `false` under Solaris.

Example:

The News Index and News Viewer portlets use content caching and can be used as a reference. The JSPs are located in the `server/public_html/portals/repository/portlets` directory in the `news_index.jsp`, `news_viewer.jsp` and `content_titlebar.jsp` files.

readOnly Content Tag

The `<cm:select>` and `<cm:selectById>` tags support an optional `readOnly` parameter.

If not specified, the default value (from `ContentHelper.DEF_CONTENT_READONLY`, which is loaded from the `commerce.content.defaultReadOnly` property in the `weblogiccommerce.properties` file) is used. Additionally, a `ContentHelper.getContent()` method and a `ContentManager.getContent()` method take a `readOnly` flag.

In the reference implementation, invoking the `getContent()` method without the `readOnly` parameter invokes the new `getContent()` method, passing in `ContentHelper.DEF_CONTENT_READONLY`. If `readOnly` is `true`, then non-EJB instances of `Content` and `Document` objects can be returned; with the reference implementation, non-EJB instances will be returned.

Note: This can help performance and prevent deadlocks. It is highly recommended that `commerce.content.defaultReadOnly` be set to `true`, and that, if `readOnly` is specified, it be specified `true`.

Object Interfaces

The interfaces `ConfigurableEntityRemote`, `ContentRemote`, `DocumentRemote`, `UserRemote` and `GroupRemote` extend both `EJBObject` and their respective non-EJBObject interfaces. For example:

```
ConfigurableEntityRemote extends EJBObject and ConfigurableEntity.  
DocumentRemote extends ContentRemote, which extends  
ConfigurableEntityRemote.  
UserRemote and GroupRemote extend ConfigurableEntityRemote.
```

In this fashion, `SessionBeans`, tags and utility methods can return either lightweight objects or the EJB instances, depending upon usage and parameters.

Note: In general, you should only typecast to the non-EJB interfaces (that is, `ConfigurableEntity`, *not* `ConfigurableEntityRemote`).

6 Creating and Managing Rules

Rules Management forms a key part of the personalization process by prescribing custom content to fit individual user profiles. The business logic encompassed by these rules allows robust delivery of personalized content marketed specifically to each end-user type.

This topic includes the following sections:

- What Is the Rules Manager?
 - Well-known Objects
 - How the Rules Engine Works
 - What Are Rule Sets?
 - Classifier Rules
 - Content Selector Rules
 - Debugging Rule Sets
- Using the Rules Management Administration Tool
 - Creating a Rule Set
 - Opening a Rule Set
 - Editing Rule Set Properties
 - Saving a Rule Set
 - Deleting a Rule Set
 - Navigating Between Rule Types

- Finding a Rule
- Creating a Classifier Rule
- Editing a Rule
- Editing Rule Properties
- Adding an If User Phrase to a Rule
- Editing an If User Phrase
- Deleting a Rule Phrase
- Creating a Content Selector Rule
- Adding an If User Classifier to a Content Selector Rule Phrase
- Adding an And When Phrase to a Content Selector Rule
- Adding a Then Display Content Phrase to a Content Selector Rule
- Editing a Then Display Content Phrase

What Is the Rules Manager?

WebLogic Personalization Server offers a robust personalization solution through a set of components that provide edit-time and run-time services for delivering personalized content to end-users while browsing a Web site. These personalization components use business rules to match users and groups with appropriate content. The logic encompassed by the rules forms a critical piece of the personalization process.

The Rules Management component of WebLogic Personalization Server provides editing, deploying, and run-time capabilities for providing personalized content based on externalized rules. This component includes two major parts: an edit-time tool with a graphical user interface (GUI) that allows developers to define classification and content selection rules, and a run-time service that matches users with content based on these rules.

Well-known Objects

The Rules Management component uses several well-known objects:

- **Content:** This object is relevant to content selector rules. It corresponds to whatever content type is selected for a set of object types. For each content selector rule invocation, a single `Content` object will be provided in the editor as a shorthand way for the rule writer to specify the nature of the content query to be produced. For example, `Content.size < 10000` would specify a part of a query to find content items having a size attribute with a value less than 10,000.
- **Now:** A well-known object in the rule editor, of type `Datetime`. A valid expression might be: `Now == Jan 01, 2001, 00:00:00 MST`.
- **User:** For each call to the rules component, a single `User` object will be provided for use by the rules. `User` has a fixed schema, determined dynamically at edit-time by calling the User Management component. Given that the `User` might have a `Numeric` schema attribute called `age`, a valid expression might be: `User.age > 35`.
- **Request:** This object is used in the same way as the `User` object. The `Request` properties are defined in a default property set. (For more information, see “[Default Request Property Set](#)” in the *WebLogic Personalization Server Developer’s Guide*.)
- **Session:** This object is used in the same way as the `User` object. The `Session` properties are defined in a default property set (For more information, see the “[Default Session Property Set](#)” in the *WebLogic Personalization Server Developer’s Guide*.)

How the Rules Engine Works

The Rules Engine functions with a set of rules operating on objects in working memory. This working memory is first populated with input from the calling objects, and contains the cached user profile bean, among other things. A representation of the user’s profile exists in working memory before any rules are actually fired.

Rules can be executed only within a context. The context associates a rule set with a working memory. The context provides an interface to the Rules Engine that controls the relationship between the rule part of the application and the working memory.

This working memory is operated on by the production rules, which are contained in rule sets. The left-hand sides (LHS) of these rules are evaluated against the objects in the working memory. If the patterns on the LHS are matched, then the actions contained in the right-hand side (RHS) of the rules are performed. Some of these actions may assert new objects into the working memory. For example, if our Classifier rule tests for `USER.age > 45`, then we might assert a new Classifier object into working memory.

The production system is executed by performing the following operations:

1. **Match:** Evaluates the LHSs of the rules to determine which are satisfied given the current contents of working memory.
2. **Conflict resolution:** Selects one rule with a satisfied LHS. If no rules have satisfied the LHSs, halts the interpreter.
3. **Act:** Performs the actions in the RHS of the selected rule.
4. **Go to step 1.**

Rules will continue to operate on the working memory until the conflict resolution set is zero (that is, no more rules can fire).

After the Rules Engine has halted, the rules service component returns a list of objects remaining in working memory. A likely scenario will have an object remaining of the type “Classification” or “ContentQuery.”

The RulesServiceBean will then iterate over these remaining objects and filter first with the input object types (there are “Classifier” and “ContentQuery”) and then optionally with the name of the rule that was requested to fire in the first place.

The resultant objects, if any, are then returned to the Advisor. As the Advisor expects to return a Boolean to the calling agent, the existence of an object determines the result of the advislet’s request.

What Are Rule Sets?

The BEA WebLogic Personalization Server provides rule sets that comprise a set of classifier and content selector rules. These rule sets act as containers for rules that match personalized content with users.

A saved rule set generally contains a set of related rules. Rules within a rule set may refer to any properties. In general, you should not change or delete properties if a rule refers to it. Adding properties does not affect existing rules.

Note: In some cases, the Rules Engine will need to evaluate each rule in a rule set, so for the sake of performance, it is a good idea to keep each rule set small.

Classifier Rules

Classifier rules dynamically categorize users into groups (user segments) using simple boolean logic. A classifier rule determines if a user profile meets a set of conditions and places the user in a category based upon the result. Essentially, if the user profile meets the conditions, it is classified according to the classifier rule; if it does not meet the classification conditions, the user profile is not included in the classification group.

The following examples illustrate the logic involved in processing a classifier rule (note the implicit *and* between the rule phrases):

```
Classifier MiddleAgeMan
If User has the following characteristics:
    User.age > 35
    and User.age < 65
    and User.gender == "M" OR "male"
```

```
Classifier HighEarner
If User has the following characteristics:
    User.income > 100000
```

Classifier rules are the building blocks of more complicated rules. Content selector rules can use classifier rules as they select personalized content to match a user or group profile. (For more information, see [“Content Selector Rules”](#) on page 6-8.)

Use the `<pz:div>` JSP tag to include a classifier rule in a JSP page. (For more information, see `<pz:div>` in the chapter [“JSP Tag Library Reference”](#) in the WebLogic Personalization Server *Developer’s Guide*.)

Adding “and” conditions to a classifier rule

To add “and” conditions to a classifier rule:

1. From the Administration Tools Home page, click the Rules Management icon.

2. Click the name of the ruleset that contains the rule.
3. Click the name of the rule to which you want to add a phrase. (This rule already has at least one phrase.)
4. Click **Phrase** to launch the Create If Phrase Wizard.
5. Select the **Single-Value With Constant** radiobutton, then click **Next**.
6. From the **Properties** list, select a property used for the rule, then click **Next**.
7. Select a comparator and enter a value to compare against the selected property.
8. Click **Or** if you want other conditions to evaluate against the same property, then click **Save**.
9. The **Edit Rule** page appears, with the rule displayed as a link below the first rule, joined by an "and."

Example:

The following example produces a rule that says:

"If the customer is a rock climber and lives in Denver, then show message about a sale at Mountain Sports."

The classifier rule would look like this:

```
USER.Customer Properties.CustomerType==Climber  
and USER.Customer Properties.ContactAddressCity==Denver
```

The jsp page would look something like this:

```
<pz:div  
  
ruleSet="jdbc://com.beasys.commerce.axiom.reasoning.rules.RuleShe  
etDefinitionHome/ShopperRules"  
rule="DenverClimber"  
id="classifier">  
  
There's a sale going on at Mountain Sports this weekend!<br>  
</pz:div>
```

Creating Classifier Rules Based on Request Properties

To create classifier rules based on request properties:

First, use the Administration Tool to create the request property.

1. Click the Property Set Management icon. The Property Set view page appears.
2. Create a request property set
or
Select an existing request property set to which you would like to add a property.
3. To create the property, click Create, then complete the form. (In the example to follow, we will create a property called `age`.)
4. You can now set the default value for the property, or you can set the value within the HTTP request using JSP.

Next, create the rule that uses the request property

Follow the same steps for creating a classifier rule, but select the desired request property from the drop down list in step 2 of the wizard.

Example

The following example produces a rule that says, "If this person is middle-aged, display 'Every vote counts. Get out and vote!'"

The rule: (Ruleset is called `RequestAge`, and rule is called `middleage`)

```
REQUEST.requestPropertySet.userAge>40  
and REQUEST.requestPropertySet.userAge<70
```

The jsp page:

```
<%@ taglib uri="pz.tld" prefix="pz" %>  
  
<!-- Request properties should be set/get with  
HttpRequest.setAttribute() and HttpRequest.getAttribute() -->  
  
<% Long age = new Long(52);  
request.setAttribute("age", age);%>  
  
<pz:div  
  ruleset="jdbc://com.beasys.commerce.axiom.reasoning.rules.  
  RuleSheetDefinitionHome/RequestAge"  
  rule="middleage" >  
  
  Every vote counts. Get out and vote!<br>  
</pz:div>
```

```
<%  
    // Clean up the session in case you are testing this over and  
    over request.removeAttribute("age");  
%>
```

Content Selector Rules

Content selector rules construct queries on the fly and return content based on the user profile. This type of rule adds time and content components to the basic classifier rule and may use references to classifier rules to define it. It also produces dynamic queries at runtime to select content from a document collection.

The power of producing dynamic queries that match content with user profiles allows content selectors to deliver highly customized content to end-users. Since content selector rules can use queries to select content based on run-time parameters, they allow the system to match personalized content to user profiles.

Note: Although a profile may meet the criteria of a content selector rule, the rule may not return any content objects. Why? If no content matches the query's criteria, the query cannot return a content object.

Content selector rules contain three parts. Each part contains zero to many phrases. The generic description of each rule part includes:

- *if user:* (Optional) the decision structure that determines if a profile meets a set of criteria.
- *and when:* The time component of the rule.
- *then display content based on:* Queries and selects content based on a set of criteria.

Example:

In this example, we are creating a rule that says:

“If the user is a high income middle aged man with a net worth over a million (dollars), then show him all HTML documents that are less than 20,000 (characters) with an investment_type that matches the user's investment type and investment_qualities that match the user's risk preference.” (This requires that “investment_type” and “investment_qualities” are fields in the content metadata.)

The following example demonstrates the prototype for a content selector rule:

```
ContentSelector JanuaryStockQuotes (type: StockQuote)
If user has the following characteristics:
  Classifier MiddleAgeMan
  and Classifier HighEarner
  and User.net_worth > 1000000

And when:
  Now >= "Jan 01, 2001, 00:00:00 MST"
  and Now < " Feb 01, 2001, 00:00:00 MST"

Then select Content where:
  Content.size < 20000
  and Content.MIMEType LIKE "*html*"
  and Content.investment_type == User.investor_type;
  and ANY Content.investment_qualities == User.risk_preference
  and Content.creation_date > Now
```

Example:

The following example shows how to implement a content selector rule. We are creating a rule that says:

“Show rock climbers the climbing gear that corresponds to their climbing ability.”

First, create the property set and the properties that the rule uses. The following table provides an example of the User Property Set "Climbers".

Property	Value
name	Cindy Climber
climber	true
ability	moderate

Next, create the rule set and rule for "Climbers". In this example, the rule set is "ClimberRules" and the rule name is "GearDisplay".

```
If User.Climbers.climber == true
Then select Content where:
Content.climberClassification == User.ability
```

Finally, we use the rule with JSP tags in the *.jsp page:

```
<!-- Info only -- %>
<um:getProperty propertySet="Climbers" propertyName="ability"
id="climberAbility" />
<um:getProperty propertySet="Climbers" propertyName="name"
id="climberName" />
<p> Climber <%=climberName%> climbs at a <%=climberAbility%> level.

<!-- Now display content geared towards their climbing ability --%>
<pz:contentselector id="stuff"
ruleset="jdbc://com.beasys.commerce.axiom.reasoning.rules.
RuleSheetDefinitionHome/ClimberRules"
rule="GearDisplay"
contenthome="com.beasys.commerce.axiom.document.DocumentManager"
max="-1" />

<es:forEachInArray id="product" array="<%=stuff%>"
type="com.beasys.commerce.axiom.content.Content">
<p>
  <ul>
    <li><cm:printProperty id="product" name="productName"
encode="html"
/>
    <li><cm:printProperty id="category" name="productCategory"
encode="html" />
  </ul>
</es:forEachInArray>
```

Note: Once you define a content selector rule and the content type it uses, you cannot change the content type.

Note: Use the `<pz:contentSelector>` JSP tag to include content selector rules in JSP pages. (See `<pz:contentSelector>` in the chapter “[JSP Tag Library Reference](#)” in the *WebLogic Personalization Server Developer’s Guide*.)

Debugging Rule Sets

You might notice that a rule set you have used in the past begins functioning incorrectly. This behavior is probably due to a change in the property set with which the rule set has a relationship.

What Is the Relationship Between Property Sets and Rules?

Rules rely on property sets to provide the properties they use to evaluate user and group profiles. If a property is modified after a rule that uses it has been created, rules may contain dangling references to properties that no longer exist or that have been changed.

As much as possible, you should avoid modifying properties after defining rules that rely upon them. Since the property set defines the schema for the properties the rules act upon, any change to the properties the rules use will affect the schema and may alter the validity of the rules. In general, be careful when modifying or deleting existing properties.

Note: You can add properties without affecting existing rules.

Content Type and Content Selector Rules

Another problem can occur when you change a content's metadata types after creating a content selector rule based on that content type's metadata. Remember that the content selector rule relies upon metadata to locate content. If you change content metadata and a content selector rule references the previous metadata, the rule will not work correctly.

Using the Rules Management Administration Tool

The Rules Management Administration Tool is an edit-time tool that provides an HTML-based GUI for creating, editing, and deleting rule sets.



The run-time rules service uses rule sets to create personalized content for end-users. The rule sets created via the Rules Management Administration Tools interface produce output that is stored in the database and evaluated at runtime.

Note: You should already have defined at least one property set before creating rule sets. Property sets provide the template that defines the parameters that rules act upon at runtime.

Creating a Rule Set

To create a rule set:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. On the Rules Management Home page, click Create. The Create a Rule Set edit page appears.
3. Enter a name for the rule set in the Name field.

Note: You must populate all fields that have a red star to their right before you can create the rule set.

4. Enter a description for the rule set in the Description field.
5. Click **Create** to save the rule set with the parameters you have selected. WebLogic Personalization Server saves the new rule set and returns to the Rules Management Home page.

Note: Clicking **Back** cancels the operation and returns you to the Rules Management Home page without creating a new rule set.

Opening a Rule Set

To open a rule set:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. On the Rules Management Home page, click the name of the rule set to open. The Rule Set view page appears, displaying rule set information and a list of the classifier and content selector rules included in the rule set.

Editing Rule Set Properties

To edit rule set properties:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set you want to edit.
3. On the Rule Set view page, click **Edit** in the Definition bar. The Edit Rule Set edit page appears.
4. Enter a new name and description.
5. Click **Save** to commit the changes or **Back** to cancel the changes. If you save the changes, the Rule Set view page appears with the new information displayed.

Saving a Rule Set

To save a rule set:

1. On the Rule Set view page, click Finished. The rule set is saved with the changes you made and WebLogic Personalization Server returns to the Rules Management Home page.

Note: This is the only time the rule set is saved, so ensure you click Finished before closing the rule set.

Deleting a Rule Set

To delete a rule set:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. On the Rules Management Home page, click Delete.
3. Select the name of the rule set to delete from the Rule Set Name drop-down list box.
4. Click Delete to permanently delete the rule set or click Back to cancel the delete operation. The Rules Management Home page appears after the delete operation completes.

Note: You must click OK in the dialog box that appears to confirm the delete operation.

Navigating Between Rule Types

To navigate between rule types:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open a rule set. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. In the rules list, click To Content Selectors or To Classifiers to navigate to the top of the rule type sections.

Finding a Rule

To find a rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to locate. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Locate the top of the rule type list—classifier or content selector type. (For the procedure to navigate between rule types, see [“Navigating Between Rule Types” on page 6-15.](#))
4. Click the letter in the alphabet bar that matches the first letter of the rule you want to find. An underlined letter indicates that a rule name begins with that letter. The browser displays the rule names beginning with the letter you selected at the top of the browser.

Note: You can also scroll to the rule using the browser’s scroll bar.

5. Locate the rule you need. (For information about making changes to the rule, see [“Editing a Rule” on page 6-17.](#))

Creating a Classifier Rule

To create a classifier rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set in which you want to create the rule by clicking the rule set name. (For instructions on creating a rule set, see “[Creating a Rule Set](#)” on page 6-12). The Rule Set view page appears.
3. Click Create in the Classifiers bar of the Rules list. The Create a Classifier Rule edit page appears.
4. Enter the rule’s name into the Rule Name field.

Note: Rule names *cannot* include spaces and punctuation.

5. Enter a description of the rule into the Description field.
6. Click Create to save the rule in the current rule set. The Create a Classifier Rule edit page refreshes and displays a message about the rule creation’s success or failure.
7. Create as many more rules as you need. Click Back to return to the Rule Set view page. The Rule Set view page appears with the new rule(s) you created displayed in the alphabetical rule list.

Administration Tools [home](#) [?](#)

BEA WebLogic Personalization Server

Rule set: MyBuyBeans

Classifier Rule name [Active Athlete] has been successfully created.

Create a Classifier Rule

Enter the appropriate information and click create. Fields denoted by a red star are required.

Rule Name: *

Description:

[back](#) [create](#)

Editing a Rule

To edit a rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set that contains the rule by clicking the rule set name. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Locate the rule you want to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Edit the rule using the instructions included in this documentation.

Editing Rule Properties

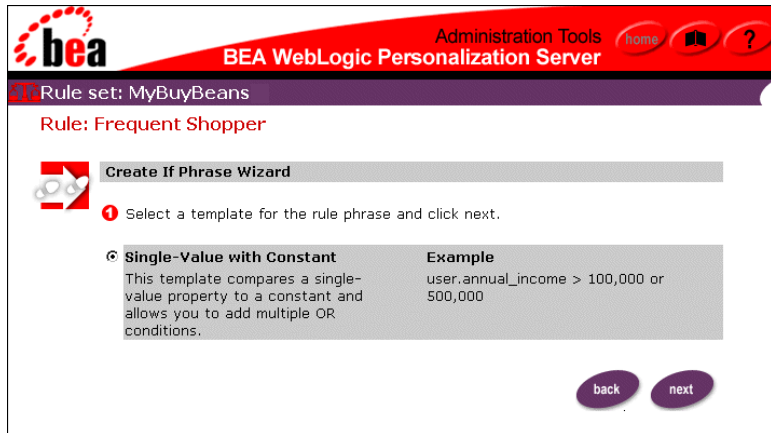
To edit rule properties:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set containing the rule you want to edit. (For more information, see [“Opening a Rule Set” on page 6-13.](#)) The Rule Set view page appears.
3. Find the appropriate rule to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click Edit in the Definition bar. The Edit Rule Definition edit page appears.
6. Enter a new name and description.
7. Click Save to commit the changes or Back to cancel the changes. The Rule view page appears with the new information displayed.

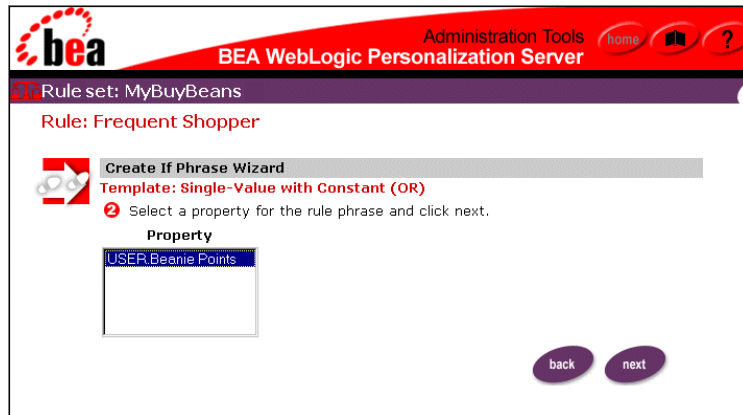
Adding an If User Phrase to a Rule

To add an If user phrase to a rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set that contains the rule to which you want to add a phrase. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule you want to modify. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click Phrase to add a phrase to the rule. Step 1 of the Create If Phrase Wizard appears.

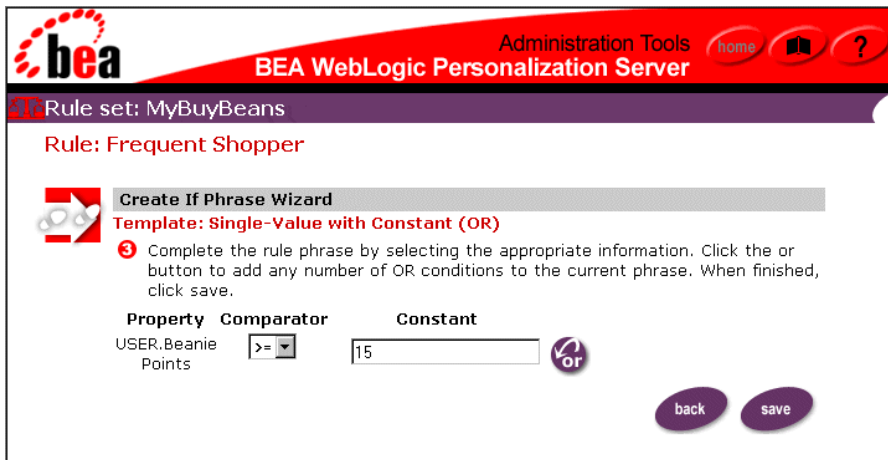


6. Select the Single-Value with Constant template to define the phrase and click Next. Step 2 of the Create If Phrase Wizard appears.



Note: Click Back in any wizard page to return to the preceding page in the Rules Management interface.

Select the property to use to define the left operand of the rule phrase and click Next. Step 3 of the Create If Phrase Wizard appears.





7. Select a comparator from the Comparator drop-down list box.

The following table shows the possible comparators for each property type:

String	Boolean	Integer	Float	Set
==	==	==	==	contains
!=	!=	!=	!=	
>		>	>	
>=		>=	>=	
<		<	<	
<=		<=	<=	

Note: You can use “and” or “or” to connect expressions that contain comparators. These are the only conjunctions or disjunctions that are allowed in expressions that contain comparators.

8. Enter a value into the Constant field.
9. To add an “or” condition to the phrase, click the  icon. The wizard page adds another Comparator drop-down list box and Constant field to the phrase below the current Comparator drop-down list box and Constant field.
10. Select a comparator and enter a value into the new Constant field. Click the  icon again to add any number of conditions to the phrase.
11. When you have completed construction of the phrase, click Save to add the phrase to the rule or Back to return to the previous step of the Create If Phrase Wizard.

Note: You can create “and” conditions by creating multiple phrases for the same rule.

Editing an If User Phrase


To edit an If user phrase:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to edit. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click one of the phrases listed below the If the user has the following characteristics bar. Step 3 of the Create If Phrase Wizard appears.
6. Select a comparator from the Comparator drop-down list box and enter a value into the Constant field.
7. Click Save to commit the changes or Back to cancel the changes. The Rule view page appears with the changes you made to the phrase displayed.

Deleting a Rule Phrase

To delete a rule phrase:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule to which you want to add a phrase. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule which contains the rule phrase to delete. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.

5. To delete a phrase, click the  icon next to the phrase name. After you confirm the delete process, the page refreshes and the new page does not show the phrase you deleted.

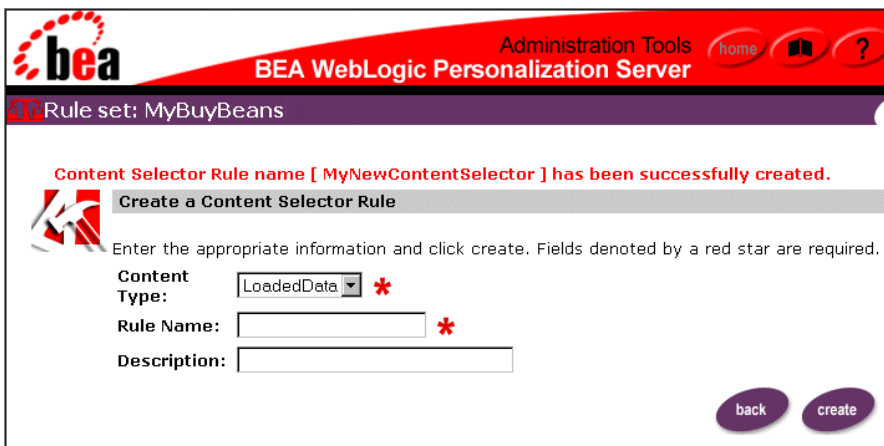
Creating a Content Selector Rule

To create a content selector rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set in which you want to create the rule by clicking the rule set name. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Click Create in the Content Selectors bar of the Rules list. The Create a Content Selector Rule edit page appears.
4. Select the content type from the Content Type drop-down list box.
5. Enter the rule’s name into the Rule Name field.

Note: Rule names *cannot* include spaces and punctuation.

6. Enter a description of the rule into the Description field.
7. Click Create to save the rule in the current rule set. The Create a Content Selector Rule edit page refreshes and displays a message about the rule creation’s success or failure.
8. Create as many rules as you need. Click Back to return to the Rule Set view page. The Rule Set view page appears with the newly created rule(s) displayed in the alphabetical rule list.



Adding an If User Classifier to a Content Selector Rule Phrase

To add an If user classifier to a Content selector rule phrase:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to edit. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click Classifier in the If the user has the following characteristics bar. The Rule search page appears.
6. Enter the name of the rule in the Classifier Name field and click Search to find a rule or click a letter to view all classifier rules that begin with the letter you click. Rules matching the search criteria populate the page when the search completes.

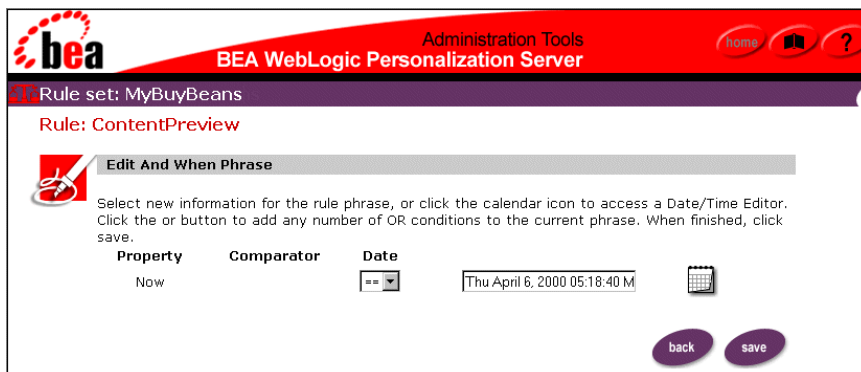
Note: The * character allows you to perform a wildcard search. Using the * character alone returns a list of all classifier rules.

7. Check the boxes next to the classifier you want to add to the rule.
8. Click Save to commit the changes. The Rule search page refreshes and displays a message about the process's success or failure.
9. Add as many classifier rules as you need. When you finish, click Back to return to the Rule view page. The Rule view page appears with the new classifier(s) displayed beneath the *If the user has the following characteristics* bar.


Adding an And When Phrase to a Content Selector Rule

To add an And When phrase to a Content selector rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to edit. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click Phrase in the And when bar. The Edit And When Phrase edit page appears.



6. Select a comparator from the Comparator drop-down list box.

7. Click the  icon to display the calendar, then select a date and time to compare. Click Save within the calendar window.

Note: You cannot enter a date by typing into the date field. The field only accepts the date string in ‘MM/dd/yyyy HH:mm:ss z’ format.

8. Click Save to add the phrase to the rule or click Back to cancel the phrase creation and leave the rule as it was before. The Rule view page appears and displays the new phrase.

Adding a Then Display Content Phrase to a Content Selector Rule

To add a Then Display Content phrase to a Content selector rule:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to edit. (For instructions on creating a rule set, see [“Creating a Rule Set”](#) on page 6-12.) The Rule Set view page appears.

3. Find the rule to edit. (For information on locating a rule, see “Finding a Rule” on page 6-15.)
4. Click the name of the rule. The Rule view page appears.
5. Click Phrase in the Then display content based on bar. Step 1 of the Create Then Phrase Wizard appears.

The screenshot shows the BEA WebLogic Personalization Server Administration Tools interface. The top navigation bar is red and contains the BEA logo, the text 'Administration Tools BEA WebLogic Personalization Server', and buttons for 'home', a book icon, and a question mark. Below the navigation bar, the breadcrumb path is 'Rule set: MyBuyBeans' and the current rule is 'Rule: ContentPreview'. The main content area is titled 'Create Then Phrase Wizard' and contains a numbered instruction: '1 Select one of the following rule phrase templates and click next.' There are three radio button options, each with a description and an example:

Option	Description	Example
<input checked="" type="radio"/> Value with Constant	This template compares a single-value content property to a constant.	content.investor_type equals "high risk" risk"
<input type="radio"/> Value with Property	This template compares a single-value content property to another single-value property.	content.investor_type equals user.investor_type
<input type="radio"/> Value with Date	This template compares a single content property to a time expression.	content.creation_date > Now

At the bottom right of the wizard, there are two buttons: 'back' and 'next'.

6. Select a template to use to define the content query and click Next. Step 2 of the Create Then Phrase Wizard appears.
7. Select a property from the Property list and click Next. Step 3 of the Create Then Phrase Wizard appears.

Note: The list receives its data from the metadata stored in the document management system.

8. Select a comparator and one of the following, depending on the template:
 - Enter a constant value if you selected the Value with Constant template.
 - Select a property from the Property drop-down list if you selected the Value with Property template.

- You cannot change the value of *now* if you selected the Value with Date template.

Editing a Then Display Content Phrase

To edit a Then Display Content phrase:

1. From the Administration Tools Home page, click the Rules Management icon. The Rules Management Home page appears.
2. Open the rule set which contains the rule you want to edit. (For instructions on creating a rule set, see [“Creating a Rule Set” on page 6-12.](#)) The Rule Set view page appears.
3. Find the rule to edit. (For information on locating a rule, see [“Finding a Rule” on page 6-15.](#))
4. Click the name of the rule. The Rule view page appears.
5. Click one of the phrases listed below the phrase bar. Step 3 of the Create Then Phrase Wizard appears.
6. Select a comparator from the Comparator drop-down list. If the content selector template is “Value with Constant,” enter a value into the Constant field.
7. Click Save to commit the changes or Back to cancel the changes. The Rule view page appears with the changes you made to the phrase displayed.

Index

A

- adding
 - And phrase to rule 6-24
 - group attribute 4-41
 - If classifier 6-23
 - If phrase to rule 6-18
 - portlet to group 2-25
 - portlet to system 2-20
 - Then phrase to rule 6-25
 - user to group 4-29
- And phrase, adding to rule 6-24
- anonymous user profile 4-22
- application initialization (property set) 3-4
- associating
 - group with portal 2-23
 - portlet with demo portal 2-36
 - user with group 4-3
- attribute
 - adding for group 4-41
 - deleting for user 4-41
 - editing 2-21
 - registering for group 4-40
 - unregistering for group 4-42
- authenticating user 4-2

B

- building demo portal component 2-31
- BulkLoader 5-18

C

- classifier rule
 - creating 6-16
 - introduction 6-5
- classifying user 1-3
- colors
 - editing for demo portal 2-38
 - editing for group 2-27
 - editing for portal 2-22
- comparison operators in query 5-17
- component, demo portal 2-31
- ConfigurableEntity 3-6
- configuring
 - Content Management system 5-8
 - Document Schema EJB 5-9
 - DocumentManager EJB 5-10
- connection pool
 - example 5-12
 - setting up 5-11
- constructing query 5-5
- contact information xi
- content
 - loading with BulkLoader 5-18
 - managing
 - (versus document management) 5-5
 - managing (overview) 1-5
 - managing (property set) 3-4
 - personalizing 1-3
- Content Management system
 - configuring 5-8
 - description 5-2

Content object 6-3
content selector rule 6-8
 creating 6-22
creating
 content selector rule 6-22
 group 4-27
 portal, details 2-18
 portlet 2-14
 portlet for demo portal 2-32
 property set, procedure 3-8
 property within property set 3-9
 rulesheet 6-12
 unified profile 4-37
 user 4-33
creating classifier rule 6-16
customer support xi

D

database
 deleting group 4-45
 deleting user record 4-47
debugging rulesheet 6-10
definitions, editing 2-20
deleting
 group 4-28
 group from database 4-45
 portal 2-24
 portlet 2-17
 property 3-13
 property set 3-13
 record from database 4-47
 rule phrase 6-21
 rulesheet 6-14
 unified profile 4-39
 user 4-36
 user attributes 4-41
demo portal
 associating portlet 2-36
 building component 2-31
 creating portlet 2-32

 editing colors 2-38
 editing layout 2-37
 testing 2-39
Destination Determiner
 setting parameters for portal or
 application 2-7
Destination Handler
 setting parameters for portal or
 application 2-7
document content, querying 5-14
Document Schema EJB, configuring 5-9
document servlet 5-6
documentation, where to find it x
DocumentManager EJB, configuring 5-10
dynamic query 1-3

E

editing
 demo portal colors 2-38
 demo portal layout 2-37
 group property 4-32
 If phrase 6-21
 portal 2-19
 portal colors 2-22
 portal definitions 2-20
 portal group 2-25
 portal group colors 2-27
 portal group layout 2-26
 portal layout 2-22
 portlet 2-16
 portlet attribute 2-21
 property set 3-12
 property within property set 3-12
 rule 6-17
 rule property 6-17
 rulesheet property 6-13
 Then phrase 6-27
 unified profile 4-39
 user property 4-34
embedded rules engine 1-3

engine, embedded rules 1-3

F

finding rule 6-15

Flow Manager

Application Init property set type 3-4

G

group

adding attribute 4-41

adding portlet 2-25

adding user 4-29

associating with portal 2-23

associating with user 4-3

creating 4-27

deleting 4-28

deleting from database 4-45

editing 2-25

editing property 4-32

managing 2-25

mapping 4-45

registering attribute 4-40

removing portlet 2-25

removing user 4-31

selecting 4-44

unregistering attribute 4-42

Group component 4-3

group profile property set 3-3

H

HTTP request property set 3-3

HTTP session property set 3-3

I

If classifier, adding to rule 6-23

If phrase

adding to rule 6-18

editing 6-21

Internationalization

non-ASCII characters 5-16

J

JSP tags 5-8

L

layout

editing for demo portal 2-37

editing for portal 2-22

editing group 2-26

LDAP, viewing settings 4-43

loading

content with BulkLoader 5-18

logging on, Portal Administration Tool 2-5

M

managing

content 1-5

portal (details) 2-18

portal (overview) 1-6

portal group 2-25

portlet 2-12

property set 1-5

rule (details) 6-2

rule (overview) 1-4

user 4-1

user profile 4-3

mapping groups 4-45

N

navigating rule type 6-15

Now object 6-3

O

object

Content 6-3

- Now 6-3
- Request 6-3
- Session 6-3
- User 6-3
- opening rulesheet 6-13

P

- Personalization Advisor 1-3

- Personalization Server 1-2

- personalizing

- content 1-3

- portal

- associating group 2-23

- controlling access 2-6

- deleting 2-24

- editing 2-19

- editing colors 2-22

- editing definitions 2-20

- editing layout 2-22

- in-a-box, running 2-30

- managing (details) 2-18

- managing (overview) 1-6

- service manager 2-6

- setting parameters 2-7

- setting up software 2-5

- versus portlet 2-3

- Portal Administration Tool

- logging on 2-5

- using 2-11

- portal creation

- details 2-18

- Portal Framework

- defined 2-3

- portal group

- editing 2-25

- editing colors 2-27

- editing layout 2-26

- managing 2-25

- portlet

- adding to group 2-25

- adding to system 2-20

- associating with demo portal 2-36

- creating 2-14

- creating for demo portal 2-32

- deleting 2-17

- editing 2-16

- editing attribute 2-21

- managing 2-12

- removing from group 2-25

- removing from system 2-20

- versus portal 2-3

- printing product documentation xi

- profile

- creating (unified) 4-37

- deleting (unified) 4-39

- editing (unified) 4-39

- for user 4-5

- property set 3-3

- user (anonymous) 4-22

- profile management 4-3

- property

- creating within property set 3-9

- deleting 3-13

- editing for group 4-32

- editing for rule 6-17

- editing for rulesheet 6-13

- editing for user 4-34

- editing within property set 3-12

- property set

- and rulesheet 6-11

- application initialization 3-4

- content management 3-4

- creating 3-8

- creating property 3-9

- deleting 3-13

- editing 3-12

- editing property 3-12

- HTTP request 3-3

- HTTP session 3-3

- management, overview 1-5

- overview 3-2

- user and group profile 3-3
- property sets
 - creating a new one 2-6
- property value, retrieving 3-6

Q

- query
 - comparison operators 5-17
 - constructing 5-5
 - structuring 5-15
- querying
 - document content 5-14
 - dynamically 1-3

R

- realm
 - mapping group 4-45
 - selecting group 4-44
 - WebLogic 4-20
- record, deleting from database 4-47
- registering group attribute 4-40
- removing
 - portlet from group 2-25
 - portlet from system 2-20
 - user from group 4-31
- request
 - property set 3-3
- Request object 6-3
- retrieving
 - property value 3-6
- rule
 - adding And phrase 6-24
 - adding If classifier 6-23
 - adding If phrase 6-18
 - adding Then phrase 6-25
 - classifier 6-5
 - content selector 6-8
 - creating classifier rule 6-16
 - creating content selector rule 6-22

- deleting phrase 6-21
- editing 6-17
- editing If phrase 6-21
- editing property 6-17
- editing Then phrase 6-27
- finding 6-15
- navigating between types 6-15
- rule managing 1-4
- Rules Management component 6-2
- Rules Management tool 6-12
- rulesheet
 - and property set 6-11
 - creating 6-12
 - debugging 6-10
 - deleting 6-14
 - description 6-4
 - editing property 6-13
 - opening 6-13
 - saving 6-14

S

- saving rulesheet 6-14
- selecting group 4-44
- servlet, document 5-6
- session
 - property set 3-3
- Session object 6-3
- setting up
 - connection pool 5-11
 - portal software 2-5
- Show Document servlet 5-14
- structuring query 5-15
- support, technical xi

T

- tags, JSP 5-8
- testing
 - demo portal 2-39
- Then phrase

adding to rule 6-25

editing 6-27

tool

Rules Management 6-12

User Management 4-26

W

WebLogic

realm 4-20

U

unified profile

creating 4-37

deleting 4-39

editing 4-39

unregistering group attribute 4-42

user

adding to group 4-29

associating with group 4-3

authenticating 4-2

classification 1-3

creating 4-33

deleting 4-36

deleting attributes 4-41

deleting record from database 4-47

editing property 4-34

profile 4-5

profile management 4-3

profile property set 3-3

profile, anonymous 4-22

removing from group 4-31

User component 4-3

User Management system

introduction 1-3

overview 4-2

User Management tool 4-26

User object 6-3

UserManager EJB 4-24

V

viewing LDAP settings 4-43