Using the BEA WebLogic Personalization Server

BEA WebLogic Personalization Server 2.0
Document Edition 2.0
April 2000

Using the BEA WebLogic Personalization Server

Document Edition	Date	Software Version
1.0	January 2000	BEA WebLogic Personalization Server 1.7
1.1	February 2000	BEA WebLogic Personalization Server 1.7.1
2.0	April 2000	BEA WebLogic Personalization Server 2.0

Contents

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 covers the following topics:

- **Overview of the Personalization Server**: An overview of all tools within Personalization Server. See "Overview of the Personalization Server" on page 1-1.
- **Creating and Managing Portals**: Portal Management allows you to create personalized application content on the Internet. See "Creating and Managing Portals" on page 2-1.
- **Creating and Managing Property Sets**: Property Set Management allows you to create property sets, the schema of personalization attributes, and the properties that make up property sets. See "Creating and Managing Property Sets" on page 3-1.
- **Creating and Managing Users**: User Management joins enterprise data about users with profile data that is used to personalize the users' view of the application. See "Creating and Managing Users" on page 4-1.
- **Creating and Managing Content**: The Content Management component provides content and document management capabilities for use in personalization services. You can use the document management system (DMS) provided as a reference implementation with WLPS **Product Version:** 2.0, or you can use a third-party vendor's DMS, including Interwoven TeamSite/OpenDeploy product and Documentum 4i product. See "Creating and Managing Content" on page 5-1.
- **Creating and Managing Rules**: The Rules Management component allows developers to create business rules that turn on and off content and match

content to users according to user profile information. See “Creating and Managing Rules” on page 6-1.

What You Need to Know

This document is intended for business analysts, Web developers, and Web site administrators involved in setting up an eCommerce site using BEA WebLogic 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.
- *Application assembler*, *system analyst*, or *systems integrator* writes rules, writes, schemas, and monitors usage.
- *System administrator* installs, configures, deploys, and monitors the Web application server.
- *Java developer* extend or modifies the Enterprise Java Bean (EJB) components that make up the Commerce Server engine, if that level of customization is needed.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at http://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 2.0 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.
italics	Indicates emphasis or book titles.

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

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

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

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

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

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

1 Overview of the Personalization Server

This overview covers the following topics:

What is the Personalization Server?

Property Set Management

Personalization Advisor

User Management

Content Management

Rules Management

Portal Management

What is the Personalization Server?

The Personalization Server is a complete solution for building personalized e-commerce sites.

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 the 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. The Personalization Server has a built-in rules editor that you use with Java Server Pages (JSP) tags to deliver a responsive, customized experience for users.

With the 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.

The 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).

The 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.

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 User Management and Rules 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 a 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 on Property Sets, see **Creating and Managing Property Sets**.

Personalization Advisor

The Personalization Advisor provides content personalization capabilities by using 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 Personalization 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 personalization 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 Personalization Advisor, see the Personalization Advisor documentation.

User Management

User Management for the product is provided through a set of tools in our administration application and alternatively through any WebLogic Server Realm. Typically, new sites that rely on self-registration will use the WLPS 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 “Creating and Managing Users” on page 4-1.

Content Management

Content 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 the Personalization Server.

The Personalization Server provides a content management component. 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 content types are determined by the developer, as are the metadata properties that will be associated with the content type. The property set management scheme is not used to manage content properties, because the content management system is responsible for that functionality. A standard set of metadata properties, such as author, creation date, and MIME type, are automatically associated with all content types.

A set of object types is provided 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 “Creating and Managing Content” on page 5-1.

Rules Management

A rule can be thought of as a stylized if-then construct. In the 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 what 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 the Personalization Server are organized and saved in rulesheets. A rulesheet 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 “Creating and Managing Rules” on page 6-1.

Portal Management

Portal Management provides an HTML windowing toolkit for 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. If the customer chooses to use the Portal product, they will develop *portlets* in JSP. These *portlets* are mini-windows on to information, content or application services available in the portal. They can be minimized, maximized in their own window, edited, and provided with help. Once built, a Portal administrator selects *portlets* for availability in their portal. Once available, a portal user controls the layout and visibility of these windows.For more information on Portal management, see “Creating and Managing Portals” on page 2-1.

2 Creating and Managing Portals

The following topics are covered here:

Introduction to Portal Development

Internet portals are a key part of many eCommerce applications. They provide an entry point to the Internet as well as value-added services such as searching and application integration. Portals can be divided into two major categories:

- MegaPortals
- Enterprise Portals

Examples of MegaPortals include Netcenter and Yahoo! These provide a window to information on the Internet and represent Business-to-Consumer web applications that can be personalized.

Enterprise portals use general-purpose applications in addition to applications specific to the enterprise or industry. These are Business-to-Business web applications, and in many cases share Business-to-Consumer functionality.

The BEA WebLogic Portal™ allows you to quickly assemble both Business-to-Consumer and Business-to-Business portals that require personalized application content on the Internet.

To take full advantage of the BEA WebLogic Portal functionality, you need to know how to:

- Run the WebLogic Server
- Create Java Server Pages (JSP)
- Set up database connections
- Set up portal service manager

The BEA WebLogic Portal is a set of Java Server Page (JSP) templates, JSP tag libraries, Enterprise Java Beans (EJB), and tools that allow:

- A Java 2 Enterprise Edition (J2EE) portal web developer to build and assemble the components for a portal page
- A portal group administrator to personalize a portal page for all the members of a group
- A portal user to personalize a portal page

The BEA WebLogic Portal enables web developers to create portal web pages and personalized application content for each portal user. The BEA WebLogic Portal uses JSPs, a part of the J2EE specification, in conjunction with a special library of JSP tags, standard HTML, Enterprise Java Beans (EJB), portal end user and the portal administration tools, and a pre-configured database to store portal component entities.

Portal applications, referred to as portlets, are JSP or HTML pages that create dynamic content that can be personalized for your portal application. This content is organized and displayed in the portal page according to the personalization information stored in the portal's personalization components.

Intelligent portals can act as tour guides to points of interest, tailored for individual user preferences. For example, 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 an overview of or 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 Personalization Server, you can quickly build powerful, dynamic portals like these, as well as static ones.

To deploy the BEA WebLogic Portal in a production environment, place the portal database on any SQL-based DBMS for which you have a Java database connectivity (JDBC) driver. SQL scripts are provided to create the necessary tables. For more information, see "Getting Started with the BEA WebLogic Portal" on page 2-9.

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.

More about Personalization Server portals and portlets

Generally, a main portal page is organized into smaller display areas. Using the Personalization Server, the portal developer can create a main page layout, with flexible methods for determining custom headers, footers, look and feel elements, and the primary content areas.

Pluggable Portlets

The most information-rich part of the main page consists of a set of portlets, laid out in columns. Each portlet is a small content area, provided to display a particular type of information. These portlets are developed especially for each portal and are written in JSP, so there is great flexibility in what can be displayed. There is a standard set of development guidelines, coupled with portal services, to ensure portal and portlets are well-behaved.

The primary way dynamic functionality of the personalization components is made available to portlets is via custom JSP tags resident in tag libraries. These tags hide much of the internal runtime complexity of the Personalization Server, presenting a small, well-defined interface to its functions. Portlets may also access certain types of personalization EJBs directly, using embedded Java to access Personalization Server functionality.

Each portlet may have a series of custom pages with specific functions associated with it, accessed via button clicks on the portlet. An edit page may make available to the user HTML input elements, in which the user can enter data on preferences specific to that portlet. A full page (or pages) version may be brought up to show an arbitrary amount of detail. A help page can be set up. The portlet may also be maximized, minimized, or floated in its own window.

Portal-to-Go

When you install the BEA WebLogic Portal, a complete demo portal is set up for you and ready to run. This ready-made portal-to-go uses the Cloudscape Database Management System (DBMS) to store the Portal Demo data. Use the portal-to-go to quick start your portal development. 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.

Portal personalization

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

The Acme Demo Portal

The BEA WebLogic Portal includes a fully operational demo portal called the Acme Portal. The 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 Demo Portal are located in the following installed product directory, `public_html/portals/repository`.

For the Cloudscape demo portal, you are ready to run this demo immediately after you complete the BEA WebLogic Portal installation.

How to create a portal

1. Create portlet applications for your portal.

   A portlet is a JSP page that represents portlet application content displayed in the portal page. The portlet JSP page is responsible for creating the content which is displayed in the portal page.The BEA WebLogic Portal provides several sample portlet applications for practice.

2. Create a new portal directory.

   a. Create a new directory in the *PORTALS* directory.

   b. Copy any new or updated JSP files into your new directory.

   c. Add a new Portal Service Manager to the `weblogic.properties` file.

   d. Use the Portal Administration Tool to create a new portal.

3. Use the Portal Administration Tool to assemble and personalize the portal.

   This is done by first adding portlets and portal groups to the portal and then personalizing them.

4. In the BEA WebLogic Portal demo pages, replace the Acme logo with your own portal logo.

   The graphic images for the demo portal are located in the installed product directory, `server/public_html/portals/repository/images`.

For more information on portals or portlets, refer to the following:

For information on creating a portal, see “Creating and Administering Portals” on page 2-26”.

For information on building and running a Demo Portal, see the “Creating a Portal Using the Demo Portal” on page 2-50.

For information on creating a portlet application, see “Creating a portlet application” on page 2-13.

Where to get more information

You may need to consult the following documentation when using the BEA WebLogic Portal:

- 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

Getting Started with the BEA WebLogic Portal

The BEA WebLogic Portal™ is used in conjunction with WebLogic Server 5.1 The JavaServer™ Pages (JSP) tag libraries, portal database, class files, and documentation for the BEA WebLogic Portal are distributed and installed by the BEA Commerce Server setup program and scripts. This document describes the steps you should follow upon completion of the installation process.

When you install the BEA WebLogic CommerceServer, a complete demo portal is set up for you and ready to run. For running the demo portal, see Running the Portal-to-Go.

For development and production systems, the BEA WebLogic Portal components require a SQL-based database to store the portal personalization data. A DBMS such as Oracle 8.0.5 database can be used to store this information. To set up a DBMS, see Running the DBMS.

Running the Portal-to-Go

When you install the BEA WebLogic Portal, a complete demo portal is set up for you and ready to run. This portal-to-go uses the Cloudscape Database Management System (DBMS) to store the Portal Demo data. Use the portal-to-go to quick start your portal development.

To start the portal-to-go demo:

1. Start the WebLogic server by executing the `StartCommerce` command file in your installation directory.
2. Open a web browser window.
3. Enter the following demo portal page URL, `http://hostname:port/exampleportal` 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 *exampleportal* is the name of the Portal Service Manager servlet for the demo portal in the `weblogic.properties` file. The installed weblogic.properties file provides defaults for the port (7601) and the Portal Service Provider (exampleportal).

Example: http://mybigbox:7601/exampleportal

You can now use the BEA WebLogic Portal Administration Tool to view the Demo Portal or assemble your own portal as described in "Creating and Administering Portals" on page 2-26.

Jar files

The following table lists the installed jar files used by the BEA WebLogic Portal. Additional jars support other Personalization Server functionality. These files can be found in the installation `lib` directory.

Table 2-1 Jar Files Used by BEA WebLogic Portal

Jar File	Description
`.\lib\wljsp.jar`	WebLogic JSP
`.\lib\esportal.jar`	BEA Portal Tag Library
`.\lib\pt_admin.jar`	BEA Portal Tag Library
`.\lib\esjsp.jar`	BEA Utility Tag Library
`.\lib\`	BEA User (um_-tags.jar)
`.\ejb\portal.jar`	BEA Portal Components
`.\ejb\axiom.jar`	BEA P13n Components
`.\ejb\bridge.jar`	BEA P13n Components
`.\ejb\foundation.jar`	BEA P13n Components

Developing Portlets

The BEA WebLogic Portal™ enables you to create your own Business-to-Business or Business-to-Consumer Internet portal solution. An integral part of any portal solution is the portlet application. This guide explains what you need to know to create a portlet application including:

- The definition of a portlet application
- The steps necessary to develop a portlet application
- Portlet examples
- The code necessary to create a portlet application

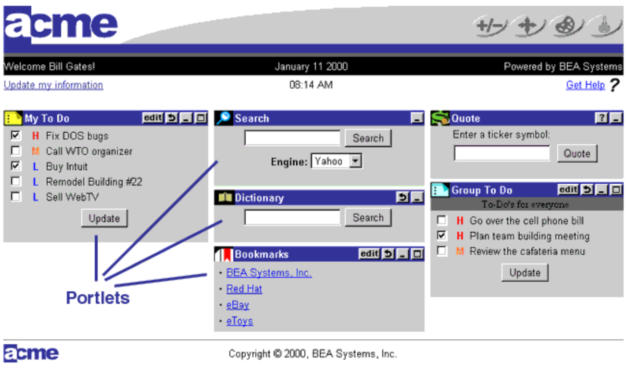
To create a portlet application, you should be a J2EE developer with a background in JavaServer Pages™ (JSP), JavaScript and HTML, and have a knowledge of Enterprise Java Beans. Also, as a portlet developer, you need to read this document to learn about the BEA WebLogic Portal framework and you should have experience with configuring and running the WebLogic Server.

What is a portlet?

From the end-user point-of-view, a portlet is a specialized content area that occupies a small 'window' in the portal page. For example, a portlet can contain travel itineraries, business news, local weather, or sports scores. The user can personalize the content, appearance, and position of the portlet according to the profile preferences set by the administrator and group to which the user belongs. The user can also edit, maximize, minimize, or float the portlet window.

The following figure shows how portlets appear in a portal home page:

Figure 2-1 Portlet Homepage View



Welcome Bill Gates! January 11 2000 Powered by BEA Systems

Update my information 08:14 AM Get Help ?

My To Do

- [✓] H Fix DOS bugs
- [ ] M Call WTO organizer
- [✓] L Buy Intuit
- [ ] L Remodel Building #22
- [ ] L Sell WebTV

Search

Engine: Yahoo

Quote

Enter a ticker symbol:

Dictionary

Bookmarks

- BEA Systems, Inc.
- Red Hat
- eBay
- eToys

Group To Do

To-Do's for everyone

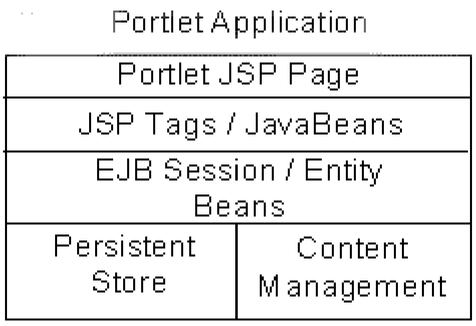
- [ ] H Go over the cell phone bill
- [✓] H Plan team building meeting
- [ ] M Review the cafateria menu

Copyright © 2000, BEA Systems, Inc.

From a server application point-of-view, a portlet is a content component implemented as a JSP that defines the static and dynamic content for a specific content subject (weather, business news, etc.) in the portal page. The portlet JSP generates dynamic HTML content from the server by accessing data entities or content adapters implemented using the J2EE platform. The Portlet JSP then displays the content in the portal.

Note: All of the portlets in a portal are included in a *single* HTML page, through the use of the `<jsp:include>` action.

Figure 2-2 Portal Application Programming Model



The diagram shown above defines the portal application programming model. This programming model includes JSP, JSP tags, JavaBeans, EJBs, data stores, and content management stores. The portlet JSP contains static HTML and JSP code. This JSP code uses application or content specific JSP tags and/or JavaBeans to access dynamic application data through EJBs, content adapters, and legacy system interfaces. Once this data is retrieved, the portlet JSP applies HTML styling to it and the generated HTML is returned in the HTTP request to the client HTTP client.

Creating a portlet application

The portlet application is a JSP that contains code responsible for retrieving personalized content and rendering it as HTML.

Once you have created your portlets, you can associate them with one or more portals. Therefore, you must create your portlet applications before using the Portal Administration Tool to create and define your portal.

Defining the Portlet JSP

The portal treats portlets as components or HTML fragments, not as entire HTML documents. The portal relies on the portlet application to create an HTML fragment for its portlet content. The portal renders the portlet's content in the portal page according to the personalization rules (the row and column position, colors, etc.) for the portal, group, and user levels.

When creating a portlet application, keep the following items in mind to ensure that your portlets run efficiently:

- Avoid using forms in a portlet that update the data within the portlet. This causes the entire portal to refresh its data which can be very time consuming. For more information on using an HTML form in a portlet, see HTML Form Processing.
- Place items that require heavy processing in an edit page or a maximized URL. Otherwise, the portal must wait for the portlet to process which considerably slows down the painting of the portal.

To define your portlet JSP:

1. Create a JSP for your portlet content.
2. Create JSPs for the portlet banner, header, footer, alternate header, alternate footer, help page, and edit URL as needed.

Note: You do not need to create a JSP for the portlet title bar because it is included in the BEA WebLogic Portal (`public.html/portals/repository/titlebar.jsp`). The portlet title bar displays the appropriate portlet titlebar icons and the name of the portlet you defined in the Portal Administration Tool.

Note: Avoid using the following HTML tags in your portlet content page. The HTML generated by the portlet content page is an HTML fragment contained in a larger portal HTML page, not a separate HTML document.

- `<html></html>`
- `<header></header>`
- `<body></body>`
- `<meta></meta>`

- `<title></title>`

3. Use the following portlet layout guidelines.

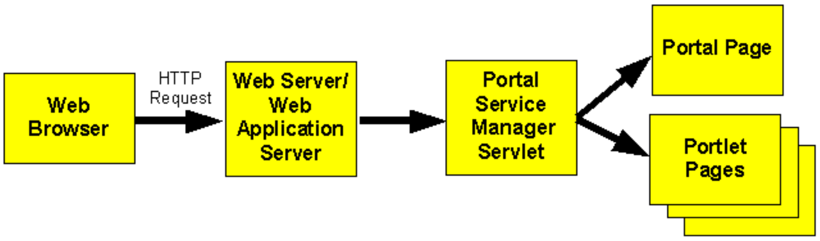
Table 2-2

Layout Attribute	**Recommendation**
Content Height	There are no restrictions on height as long as the content fits in your portal page.
Column Width	Take into account that the width of your portlet is controlled by the portal(s) it is associated with. A portal lays out your portlet content in a column based on portal, group, and user personalization rules. As a result, the width of your portlet should be well behaved.
Content Wrapping	Allow wrapping for all portlet content. Do **not** use the **NOWRAP** attribute in table cells.
Titlebar Icon Height	The image height attribute in titlebar.jsp is set to 20.
Titlebar Icon Width	The image width in titlebar.jsp is set to 27.

Working Within the Portal Framework

The portal framework consists of JavaServer Pages, JSP tag libraries, EJBs, Java servlets, and other supporting Java objects. The main Java servlet is the Portal Service Manager, referred to as the *traffic cop*. The Portal Service Manager receives all incoming HTTP requests and dispatches each request to the appropriate destination URL. As a result, all access to your portal pages is controlled by the Portal Service Manager. The following diagram shows where the Portal Service Manager fits in the portal framework.

Figure 2-3 Portal Framework


Web Browser
HTTP Request
Web Server/ Web Application Server
Portal Service Manager Servlet
Portal Page
Portlet Pages

Extending the PortalJspBase Class

It is recommended that your portlet JSP extend the framework's `PortalJspBase` Java class. This class contains many convenience methods which perform general tasks for your portlet JSP page, such as accessing session information, the *traffic uri*, and user login information.

To extend the `PortalJspBase` class, include the following code at the top of your portlet JSP:

```
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"%>
```

Accessing Portal Session Information

The portal session information you can access from the `PortalJspBase` class are listed in the following table which lists the name, type, and description for each session value. For more information, see the Portal API Documentation.

Table 2-3

Session Value Name	Type	Description
`PortalAdminConstants.PORTAL_NAME`	`String`	The name of the portal associated with the current request.
`JspConstants.SERVICEMANAGER_SUCCESSOR`	`String`	The name of the successor associated with the current session. The successor profile properties are used for those properties not specified by the user.

Table 2-3

Session Value Name	Type	Description
`JspConstants.SERVICEMANAGER_USER`	`String`	The name of the user associated with the current session.
`UserManagerConstants.PROFILE_USER`	`Configurable Entity`	The user profile associated with the current request or the session.
`UserManagerConstants.PROFILE_SUCCESSOR`	`ConfigurableEntity`	The group profile associated with the current request or the session.
`UserManagerConstantsPROFILE_SUCCESSOR_UID`	`Long`	Unique IDs for the configurable Entities.
`UserManagerConstantsPROFILE_USER_UID`	`Long`	Unique IDs for the configurable Entities.
`SERVICEMANAGER_USER`	`String`	The name of the user associated with the current request.

You can retrieve the portal session information described above through the following `PortalJspBase` methods:

- `public Object getSessionValue(String` *`aName`*`, HttpServletRequest` *`aRequest`*`)`
- `public void setSessionValue(String` *`aName`*`, Object` *`aValue`*`, HttpServletRequest` *`aRequest`*`)`
- `public void removeSessionValue(String` *`aName`*`, HttpServletRequest` *`aRequest`*`)`

You can set the portal session's `SERVICEMANAGER_USER` and `SERVICEMANAGER_SUCCESSOR` through the following `JspBase` methods:

- public static void setUser(String *`aUser`*, HttpServletRequest *`aRequest`*)
- public static void setSuccessor(String *`aSuccessor`*, HttpServletRequest *`aRequest`*)
- public static void setUserAndSuccessor(String *`aUser`*, String *`aSuccessor`*, HttpServletRequest *`aRequest`*)

Sending Requests Through the Portal Service Manager

Remember that all HTTP requests and responses are sent to the Portal Service Manager servlet. Therefore, your portlet HTML must refer to the Portal Service Manager's URL for URL links and HTML form processing.

Using URL Links in Your Portlet

If your portlet contains links to a JSP page that is not a portlet, use the following `PortalJspBase` method to create your URL and to guarantee that the HTTP request is sent to the service manager URL:

```
public String createURL(HttpServletRequest aRequest, String
destination, String parameters)
```

The destination should be a relative or qualified file location in the form such as `example/mytodo.jsp`, or `/yourportal/example/mytodo.jsp`. The path is relative to the `documentRoot`, as specified in `weblogic.properties`. Parameters should be a string such as `column=4&row=5`.

Note: Parameter values should already be encoded as you would for any HTTP request. Example: `String parms = "column=" + java.net.URLEncoder.encode("4");`

Because of the way the JSP engine handles `jsp:forward` and `jsp:include`, you must *fixup* the relative URLs in your portlet, especially relative links to images. The web browser thinks the root for relative links is the directory in which the Portal Service Manager resides and not your portlet's directory.

To fixup relative URLs use the following `ToolsJspBase` method:

```
public static String ToolsJspBase fixupRelativeURL(String aURL,
HttpServletRequest aRequest)
```

where *`aURL`* is the destination URL to fix up and *`aRequest`* is the current HTTP request. In your JSP page, use the following method to code a 'fixup':

```
<img src="<%=fixupRelativeURL("images/quote.gif",
request)%>"width="50" height="35" border="0">
```

Note: For the repository feature to work with `jsp:include` and `jsp:forward`, use *reconcile file* to determine the correct location of the file that is included or forwarded.

Example: `jsp:forward page=<reconcileFile("login.jsp")%>`

HTML Form Processing

If your portlet contains an HTML form, send all requests to the Portal Service Manager and set the destination request parameter.

To process HTML forms:

1. Set the form action to `action=getTrafficURI(request)`. This sends the form action request to the Portal Service Manager. This calls the `PortalJspBase` method:

   ```
   public String getTrafficURI(HttpServletRequest aRequest)
   ```

   The following example shows the use of the HTML form action to send a form request to the Portal Service Manager:

   ```
   <form method="post" action="<%=getTrafficURI(request)%>">
   ```

2. Set the destination request parameter in the HTTP post request. This tells the Portal Service Manager where to dispatch the request.

To set the request destination for HTML forms, enter the following code within your form in your JSP page:

```
<input type="hidden" name="<%=DESTINATION_TAG%>"
value="example/mytodo.jsp">
```

Note: Do not go through the Portal Service Manager for HTTP requests to other servers.

Retrieving the Home Page

The Portal Service Manager sets the home page for each portal in the Portal Framework session information. The home page is registered as an initial argument for Portal Service Manager servlet in weblogic.properties. Use the following `PortalJspBase` method call to retrieve the home page:

```
public String getHomePage(HttpServletRequest aRequest)
```

Retrieving the Current Page

You can also retrieve the current page from the Portal Framework session information by using the following `PortalJspBase` method:

```
public String getCurrentPage(HttpServletRequest aRequest)
```

Note: When you maximize a portlet, the current page changes to `fullscreenportlet.jsp.`

Setting the Request Destination

When routing a request through the Portal Service Manager, you must specify the destination that should receive the request. The destination can be relative to the current page (portal.jsp, full-screen portlet.jsp, etc.) or a fully qualified path from the document root.

Note: The `DESTINATION_TAG` constant is available in `PortalJspBase`.

If your portlet contains links to other portal pages, use the following PortalJspBase method to create your URL and to guarantee that the HTTP request is sent to the service manager URL:

```
public String createURL(HttpServletRequest aRequest, String
destination, String parameters)
```

The destination should be a relative or qualified file location in the form such as `example/mytodo.jsp`, or `/yourportal/example/mytodo.jsp.`

In some cases, you may need to override the request parameter used by the Portal Service Manager. For example, use an override destination if your page contains a form that needs to be validated and forwarded elsewhere after validation. Use the following `PortalJspBase` method in your JSP page:

```
public void setOverrideDestination(HttpServletRequest req, String
dest)
```

To set the request destination for HTML forms, enter the following code within your form in your JSP page:

```
<input type="hidden" name="<%=DESTINATION_TAG%>"
value="example/mytodo.jsp">
```

Tracking User Login Status

You can log the user in or out and track whether a user is currently logged in.

Use the following `PortalJspBase` method to track the user login status of a portal session:

```
public void setLoggedIn(HttpServletRequest aRequest,
HttpServletResponse aResponse, boolean aBool)
```

```
public Boolean getLoggedIn(HttpServletRequest aRequest)
```

Loading Content from an External URL

According to the JSP specification, a JSP processed by a JSP engine must be relative to the server in which the JSP engine is running, requiring that all of your portlets reside in your portal server and not on an external web site. However, you can use the `uricontent` tag to download the contents of an external URL into your portlet. If you download the contents of a URL into your portlet, you need to fully qualify the images located on the remote server because the relative links contained within the remote URL will not be found unless fully qualified.

Use the following method to load content from an external URL:

```
<es:uricontent id="uriContent"

               uri="http://www.beasys.com/index.html">
<%
out.print(uriContent);
%>

</es:uricontent>
```

```
The sample <es:uricontent> tag is available in
public_html/portals/repository/portlets/_uri_example.jsp
```

Using example portlets

The `/server/public_html/portals/repository/portlets` directory of the BEA WebLogic Portal contains example portlets. The following table lists the name of each example portlet, its description, and its associated files.

Caution: The example portlets are intended for illustration purposes only and should not be used for production code.

Table 2-4

Example Portlet	Description
`_uri_example.jsp`	Demonstrates how to implement the uricontent tag to import contents from another URL on the Internet.
`bookmarks.jsp`	Displays the bookmarks associated to the current user. ■ `bookmarks_edit.jsp` – Edit screen for the bookmarks. ■ `images/pt_bookmark.gif` – Bookmark icon for the portlet titlebar.
`definedportals.jsp`	Displays the portals defined in the system. Uses the `<es:foreachinarray>`, `<es:simplereport>`, and `<wl:sqlquery tags>`.
`definedportlets.jsp`	Displays the portlets defined in the system. Uses the `<es:foreachinarray>`, `<es:simplereport>`, and `<wl:sqlquery tags>`.
`dictionary.jsp`	Demonstrates how to redirect a portlet to an external site. ■ `images/pt_dictionary.gif` – Dictionary icon for the portlet titlebar
`generic_todo.jsp`	For a complete `generic_todo.jsp` example, see Using the Default Implementation.
news_index.jsp	Demonstrate use of `<cm:>` tags.
`news_viewer.jsp`	Display content driven from `content_index.jsp.` (Use in conjunction with `content_index.jsp`.).

Table 2-4

Example Portlet	Description
`grouptodo.jsp`	Displays a `Group To Do List`. ■ `todo.jsp` – Statically included file that does not run by itself. It requires user information from `grouptodo.jsp`. ■ `grouptodo_edit.jsp` – Edit URL for `grouptodo.jsp`. ● `todo_edit.jsp` – Statically included file that does not run by itself. It requires user information from `grouptodo_edit.jsp`. ■ `grouptodobanner.jsp` – Banner for the `grouptodo.jsp`. ■ `images/pt_group_list.gif` – Group To Do List icon for the portlet titlebar.
`mytodo.jsp`	Displays a “My To Do List.” ■ `todo.jsp` – Statically included file that does not run by itself. It requires user information from `mytodo.jsp`. ■ `mytodo_edit.jsp` – ’Edit URL’ for `mytodo.jsp`. ● `todo_edit.jsp` – Statically included file that does not run by itself. It requires user information from `mytodo_edit.jsp`. ■ `images/pt_my_list.gif` – ’My To Do List’ icon for the portlet titlebar.
`quote.jsp`	Demonstrates how to redirect a portlet to an external site. ■ `images/pt_quote.gif` – ’Quote’ icon for the portlet titlebar.

Table 2-4

Example Portlet	Description
`search.jsp`	Demonstrates how to redirect a portlet to an external site. ■ `images/pt_search.gif` – 'Search' icon for the portlet titlebar.

Portlet JSP example

The following example shows many of the defined method calls and tags mentioned in this document. Each tag is a file in the default implementation directory `portals/repository/portlets`, named `generic_todo.jsp`. The bean associated with this file is `example.portlet.bean.TodoBean`. The source of the bean is in `<install-dir>`/src/. Using these files, you can recreate each example in your JSP file.

```
<%--
Set up the tag libraries for tag references. Also have the page extends
PortalJspBase, so that you can have access to helper methods.
--%>

<%@ taglib uri="lib/esjsp.jar" prefix="es" %>
<%@ page extends="com.beasys.portal.admin.PortalJspBase"%>
<jsp:useBean id="todoBean" class="example.portlet.bean.TodoBean"
scope="request"/>
<jsp:setProperty name="todoBean" property="*"/>

<%
// Get the user name out of the session
String owner =
(String)getSessionValue(com.beasys.portal.tags.PortalTagConstants.PORTAL_USER,
request);
%>
<%-- Use the preparedstatement tag to execute a query --%>
<es:preparedstatement id="ps" sql="<%=todoBean.QUERY%>" pool="jdbcPool">
<%
todoBean.createQuery(ps, owner);
java.sql.ResultSet resultSet = ps.executeQuery();
todoBean.load(resultSet);
%>
```

```
</es:preparedstatement>
<%
String target = request.getParameter("target");
// Use this method to validate that the request that is being
// processed is actually for this jsp page.
if ( target != null && target.equals(getRequestURI(request)))
{
%>
<es:preparedstatement id="ps" SQL="<%=todoBean.UPDATE%>" pool="jdbcPool">
<%
todoBean.process(request, owner, ps);
%>
</es:preparedstatement>
<%

}

// Get the enclosing window out of the session
// Get the current page out of the session.
// For this example, this value will be the
// fullscreenportlet.jsp (with args) or portal.jsp.

String value = getCurrentPage(request);
%>
<%--
set the action on the form to send the post back to the ’traffic cop’
--%>
<form method="post" action="<%=getTrafficURI(request)%>">
<table width="100%" border="1">
<tr>
<td>
<table width="100%" border="0">
<%
String[][] results = todoBean.asTable();
%>
<%--
Use the foreachinarray tag to iterate over the query results.
--%>
<es:foreachinarray id="nextRow" array="results" type="String[]">
<TR>
<td width="10%">
<div align="center">
<input type="checkbox" <%=nextRow[0]%> name="<%=todoBean.CHECKBOX+nextRow[2]%>"
value="ON">
</div>
</td>
<td width="10%" align="center"><%=nextRow[1]%></td>
```

```
<td width="80%"> <%=nextRow[2]%></td>
</TR>
</es:foreachinarray>
</table>
</td>
</tr>
<tr>
<td>
<div align="center">
<input type="submit" name="updateButton" value="Update">
</div>
</td>
</tr>
</table>
<input type="hidden" name="owner" value="<%=owner%>">
<%--
Tell the ’traffic cop’ that you want the post to come back to this page.
--%>
<input type="hidden" name="<%=DESTINATION_TAG%>" value="<%=value%>">
<%--
Use getRequestURI(request)to get this page’s name and location.
Also use it if you want to verify that the request is for this page,
by setting a param for the target.
--%>
<input type="hidden" name="target" value="<getRequestURI(request)%>">

</form>
```

Creating and Administering Portals

The BEA WebLogic Portal Administration Tool contains a complete set of functions that enable Portal Administrators to easily create and update BEA WebLogic Portal database schema entities. With the HTML-based, graphical user interface tool, you can build and assemble the components of a portal page and personalize the portal’s content, layout, and appearance.

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

The following topics explain how to create and administer a portal using the BEA WebLogic Portal Administration Tool.

Setting Up

Before using the Portal Administration Tool to create and administer a portal, you must install and setup the BEA WebLogic Portal software.

You must also complete the following three tasks before you can to log on to and use the tool:

- Set the WebLogic Server document root
- Configure the portal service manager to control portal access
- Create a portal web site directory under the server document root

Set the WebLogic Server document root

In the `weblogic.properties` file, set a WebLogic server document root in your preferred Web publishing root directory. Example:

```
weblogic.httpd.documentRoot=yourDocumentRoot
```

Following is an example of the `weblogic.properties` file:

```
weblogic.httpd.register.exampleportal=com.beasys.commerce.portal.
admin.PortalServiceManager

weblogic.httpd.initArgs.exampleportal=\
portalname=exampleportal,\
homepage=/portals/repository/portal.jsp,\
defaultdest=/portals/repository/portal.jsp,\
workingdir=/portals/repository/,\
groupname=AcmeUsers,\

sessioncomparator=com.beasys.commerce.portal.admin.PortalSessionC
```

```
omparator,\

refreshworkingdir=120,\
repositorydir=/portals/repository/,\
timeout=99999,\
allowautologin=false
```

Configure the portal service manager to control portal access

The Portal Service Manager (PSM) controls user access to your portal. It is a Java servlet that all portal framework HTTP requests must be sent to. Among other functions, the PSM:

- Restricts unauthorized access to your portal and the JSP pages they control
- Cleans up the URL shown in the browser
- Creates a routing framework on which the program can rely

You must register an instance of this servlet in the `weblogic.properties` file for each portal you deploy. The following is a sample PSM servlet registration for a portal named *myPortal*.

```
weblogic.httpd.register.myPortal=com.beasys.portal.admin.PortalServiceManager
weblogic.httpd.initArgs.myPortal=\
portalname=myPortal,\
homepage=/portals/myPortal/portal.jsp,\
groupname=everyone,\
defaultdest=/portals/myPortal/_userlogin.jsp,\
timeout=999999,\
workingdir=/portals/myPortal/,\
allowautologin=true
refresh working dir =-1,\
repositorydir=/portal/Repository/,\
sessioncomparator=com.beasys.commerce.portal.admin.Portalsessioncomparator,\
```

The table below lists valid parameters for your initial registration of the PSM servlet.

Table 2-5 Valid Portal Service Manager Servlet Parameters

Parameter Name	Required	Description
`portalname`	Yes	The name given to the portal you created in the Portal Administration Tool. Example: Demo Portal
`homepage`	Yes	The home page JSP returned by the system in auto-login or from the portal home button. (This page is qualified from `yourDocumentRoot` as defined in the `weblogic.properties` file.) Example: `homepage=/portals/myPortal/portal.jsp`
`groupname`	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 Portal Service Managers that are alike except for the groups that they service.) This value defaults to everyone.
`defaultdest`	Yes	The default destination page JSP if there is not a valid session for the user. (This page is qualified from `yourDocumentRoot` as defined in the `weblogic.properties` file.) To display a default portal page for anonymous users, use: `defaultdest=/portals/myPortal/portal.jsp` or to force anonymous users to the login page instead of the portal page use: `defaultdest=/portals/myPortal/_userlogin.jsp`
`timeout`	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 autologin=true.

Table 2-5 Valid Portal Service Manager Servlet Parameters

Parameter Name	Required	Description
`workingdir`	Yes	The working directory JSP for the portal implementation that tells the portal framework where to find your portal pages and the BEA WebLogic Portal pages.(This page is qualified from `yourDocumentRoot` as defined in the `weblogic.properties` file.) Example: `workingdir=/portals/myPortal/`
`allowautologin`	No	Determines whether a client with valid cookies can automatically login. The default is false.
repositorydir	Yes	Location of default files, (gifs, JSP, etc.)
refresh workingdir	No	Number of seconds, defaults to -1, which means check every time.
sessioncomparator	Yes	How to determine if the session is valid.

For more information on `weblogic.properties` servlet registration, see http://www.weblogic.com/docs/admindocs/properties.html#http.

Create 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 portal ’quick start’ directory to your portal web site directory.

To copy files from the portal ’quick start’ 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. Verify that the Portal Service Manager `workingdir` property value in the `weblogic.properties` file matches the name of the web site directory to which you just copied the quick-start files. See the Quick-Start 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 2-6 'Quick-Start' Directory and Subdirectories

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

Logging On to the Portal Administration Tool

Once you have prepared the WebLogic server document root, configured the Portal Service Manager, and created web site directory, you can log on to the Portal Administration Tool.

To log on to the P13N Administration Tool:

1. Start the WebLogic server configured for portal use.
2. Access `http://hostname:port/wlpsadmin` 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 *`portaladmin`* is the name of the Portal Service Manager servlet for the Portal Administration Tool in the `weblogic.properties` file.

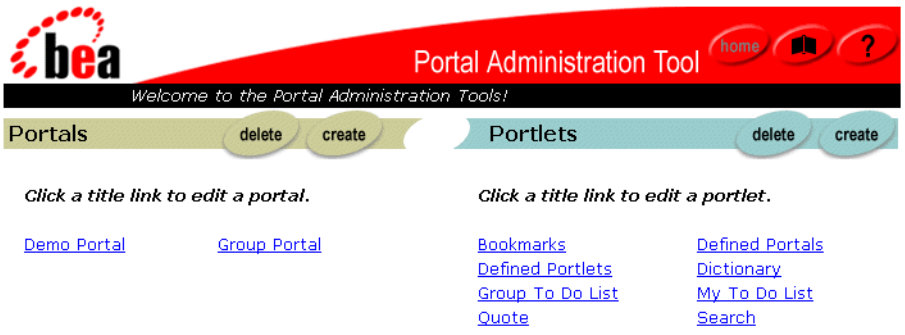
   Note: **A**ll administration tools must be accessed through the JSP Service Manager servlet.

   A dialog box appears and prompts you to enter a username and password.
3. Enter the username *system* and use the password `weblogic.password.system` property in WebLogic Commerce's `weblogic.properties` file.
4. Click **Ok** to display the P13N Administration Tool home page.
5. Click the Portal Administration page icon.

Figure 2-4 shows the Administration Tool home page.

Using the Portal Administration Tool

Figure 2-4 Administration Tool Home Page


bea
Portal Administration Tool
home
?
Welcome to the Portal Administration Tools!
Portals
delete
create
Portlets
delete
create

Click a title link to edit a portal.

Demo Portal Group Portal

Click a title link to edit a portlet.

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

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

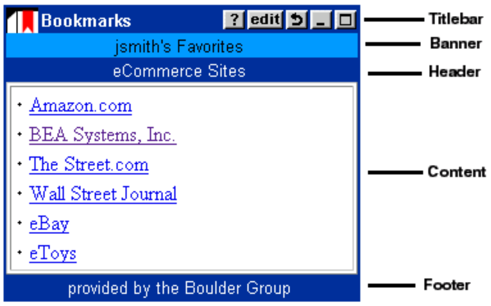
Administering portlets

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 2-5 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.

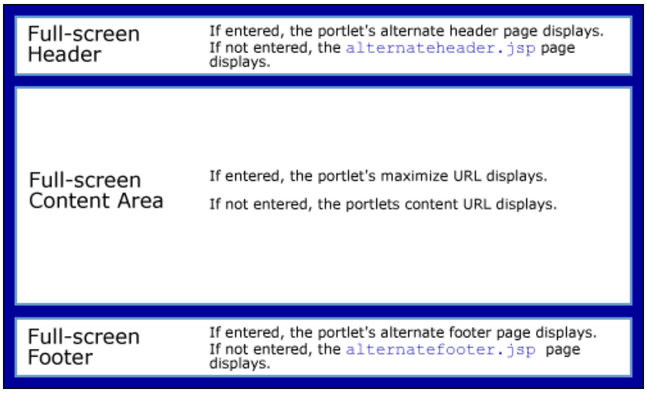
  The fullscreen page uses:

  - **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 2-6 Content Display Criteria


Full-screen Header
If entered, the portlet's alternate header page displays. If not entered, the alternateheader.jsp page displays.
Full-screen Content Area
If entered, the portlet's maximize URL displays.
If not entered, the portlets content URL displays.
Full-screen Footer
If entered, the portlet's alternate footer page displays. If not entered, the alternatefooter.jsp page displays.

- **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 Demo Portal components to quick-start your portal development, see “Creating a Portal Using the Demo Portal” on page 2-50.

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

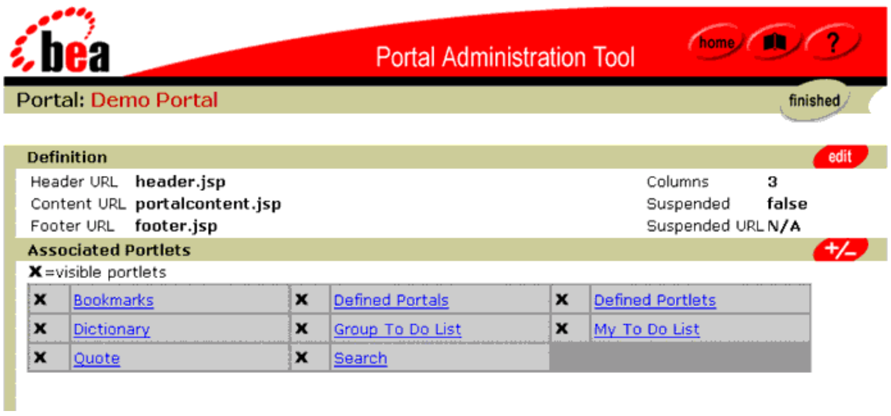
After creating a portal, you can edit it to associate portlets, groups, and users. You can also personalize the portal's layout and color scheme, and make changes to the definition.

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 2-7 Portal View Page



Portal: Demo Portal

Definition

Header URL	header.jsp	Columns	3
Content URL	portalcontent.jsp	Suspended	false
Footer URL	footer.jsp	Suspended URL	N/A

Associated Portlets

X =visible 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 redisplays 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 2-46.

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.
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.

Testing Your 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.

To test your portal:

1. In a web browser, enter the Portal Service Manager URL. Using `http://host:port/myPortal` as an example, myPortal matches the Portal Service Manager name for your portal in the `weblogic.properties` file installed in the portal directory.

   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.

Creating a Portal Using the Demo Portal

The following sections show you how the ACME Demo Portal data could be manually, entered, using the administration tools. The BEA WebLogic Portal installation package includes a Portal Service Manager for the Demo Portal. You can find it in the `weblogic.properties` file. The Portal Service Manager property name is `exampleportal` and maps to the `Demo Portal` EJB component.

Before using the Portal Administration Tool to Assemble the Acme Demo Portal, you must install and set up the BEA WebLogic Portal software. For more information, see "Getting Started with the BEA WebLogic Portal" on page 2-9.

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 "Getting Started with the BEA WebLogic Portal" on page 2-9.

For more information on accessing and using the Administration Tool, see "Logging On to the Portal Administration Tool" on page 2-31.

Building the Acme Demo Portal components

To create the 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 2-7

Field Name	Data
Portal Name	Demo Portal
Header URL	`header.jsp`
Content URL	`portalcontent.jsp`
Footer URL	`footer.jsp`
Number of columns	3
Suspend	Defaults to false. Set to true to suspend the portal during maintenance.
Suspended URL	Enter `suspended.jsp` 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, '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 2-8

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

Table 2-8

Portlet Name	Data
	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 2-9

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

Table 2-9

Portlet Name	Data
	Icon URL: `portlets/images/pt_my_list.gif`
	Minimizable: select the check box
	Floatable: select the check box
	Login Required: select the check box
My Group To Do List	**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
	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

Table 2-9

Portlet Name	Data
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 Portlets" on page 2-38.

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 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.

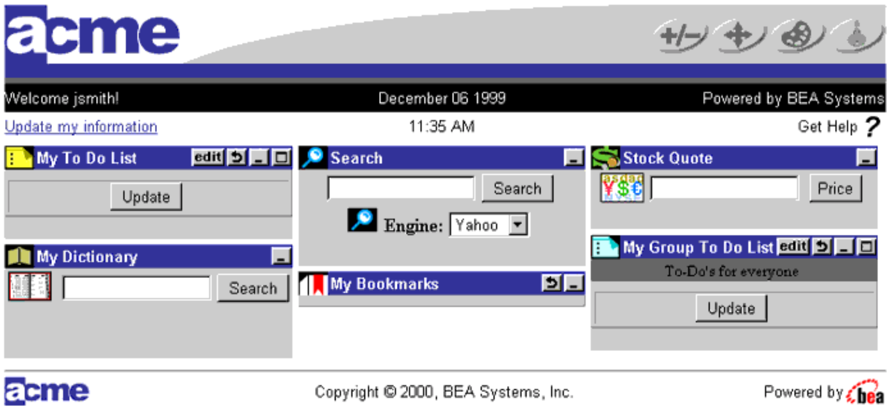
7. To save your color preferences without previewing them, click **save**. The portal view-page displays the new colors associated with the portal in the Colors section of the screen.

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 2-8 Acme Demo Portal



To test the demo portal:

1. In a web browser, enter the Portal Service Manager URL (`http://host:port/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 2-8 shows the Acme Demo Portal as it displays with the BEA default color scheme and layout to a registered user named jsmith.

BEA WebLogic Portal Framework Files

The following table displays the names and functions of the template JSP files provided with the BEA WebLogic Portal framework. Each of these files is located in the root directory of the portal which it serves, such as `/portals/repository`.

Table 2-10

JSP File Name	Function
`_user_add_portlets.jsp`	The tool employed by the end user to add/remove portlets.
`_user_layout.jsp`	The tool employed by the end user to update portlet layout.
`_userlogin.jsp`	The user login page.

Table 2-10

JSP File Name	Function
`_userreg.jsp`	The new user registration page.
`_userreg_summary.jsp`	The user profile summary page.
`alternatefooter.jsp`	The footer displayed when a portlet is maximized or detached.
`alternateheader.jsp`	The header displayed when a portlet is maximized or detached.
`baseheader.jsp`	A stripped version `header.jsp`, intended for general use beyond the portal home page.
`color_picker.jsp`	The color palette employed by the user color preferences tool.
`error.jsp`	A general-purpose page used for displaying run-time errors.
`error_footer.jsp`	The footer displayed with `error.jsp`.
`error_header.jsp`	The header displayed with `error.jsp`.
`footer.jsp`	The footer displayed with the main portal page.
`fullscreenportlet.jsp`	The page used to display a maximized or detached portlet.
`gen_prefs.jsp`	The tool employed by the end user to update general user profile information.
`header.jsp`	The header displayed with the main portal page.
`help.jsp`	The end user help page.
`layout_script.jsp`	The JavaScript used by the end user layout tool.
`portal.jsp`	The main portal page.
`portalcontent.jsp`	The page which prescribes portlet layout within the main portal page.

Table 2-10

JSP File Name	Function
`portalerror.jsp`	The default error page displayed when an access attempt to a portal page fails.
`portalnotexist.jsp`	The page which displays a general message indicated that the requested portal does not exist.
`portlet.jsp`	The page which constructs a portlet, combining portlet titlebar, banner, header, content, and footer.
`privacy_policy.jsp`	A placeholder for a company privacy policy statement.
`status.jsp`	The page used to display end-user status messages.
`suspended.jsp`	The page which provides a message indicating that the requested portal is currently non-operational, typically for maintenance reasons.
`titlebar.jsp`	The portlet titlebar. Contains appropriate portlet icons and portlet name.
`user_colors.jsp`	The end user color preferences tool.

Internationalization

The BEA WebLogic Portal™ Administration Tool is supported by JSP bean objects which employ Java internationalization standards in the practice of presenting error and status messages. These beans use a BEA utility object called MessageBundle in conjunction with text-based properties files to produce two types of locale-specific display text. The two types of text are as follow:

- Static Text
- Constructed Messages

Properties Files

Properties files are located in two particular directories in the portal framework. The first set of properties files supports the Portal Administration Tool and are located in `com/beasys/commerce/portal/admin/jspbeans/i18n`.

The second set of properties files supports both the Administration Tool and the run-time portal end-user tools. This set is located in `com/beasys/commerce/portal/jspbeans/i18n`.

Each properties file that supports a particular bean includes the bean name and a properties extension. For example, the properties file that supports the `com.beasys.portal.admin.jspbeans.PortalJspBean` bean resides in the `i18n` directory, and is called `PortalJspBean.properties`.

Static Text

The BEA WebLogic Portal uses the following convention when naming static text entries in the properties files:

```
propertyName.txt=propertyValue
```

For example: `error.txt=Error Occurred`.

A static text property is acquired from a loaded MessageBundle using the following method:

```
String messageBundle.getString(String propertyName)
```

For example:
```
System.out.printin(messageBundle.getString("error.txt"));
```

For more information, see the Portal API Documentation.

Constructed Messages

The dynamic display text created by internationalization often depends on one or more variables, and the order of these variables in a text segment is locale-specific. In this case, the BEA WebLogic Portal provides a means for constructing message segments for display.

The portal uses the following convention when naming message entries in properties files:

```
propertyName.msg=propertyValue
```

For example:

```
fieldRequired.msg={0} is a required field.
```

A constructed message is acquired from a loaded MessageBundle using the following method:

```
String messageBundle.getMessage(Object[ ] args, String propertyName)
```

For example:

```
Object[ ] args={"ContentURL"};
```

```
System.out.println(args,"fieldRequired.msg");
```

For more information, see the Portal API Documentation.

CHAPTER

3 Creating and Managing Property Sets

The following topics are covered here:

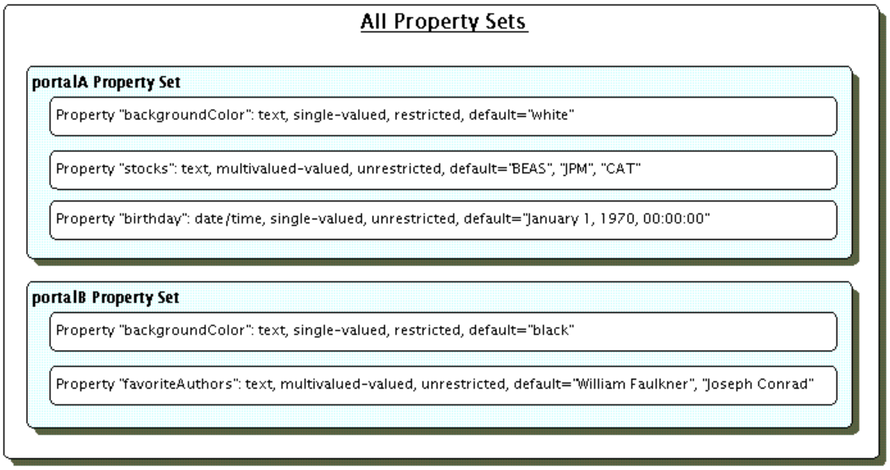
Using the Property Set Management tool, you can create property sets, the schemas for personalization attributes schema and define the properties that make up these property sets.

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, the 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 portal A and portal B. 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.


All Property Sets
portalA Property Set
Property "backgroundColor": text, single-valued, restricted, default="white"
Property "stocks": text, multivalued-valued, unrestricted, default="BEAS", "JPM", "CAT"
Property "birthday": date/time, single-valued, unrestricted, default="January 1, 1970, 00:00:00"
portalB Property Set
Property "backgroundColor": text, single-valued, restricted, default="black"
Property "favoriteAuthors": text, multivalued-valued, unrestricted, default="William Faulkner", "Joseph Conrad"

A property definition includes the following information:

- Property Value Type: The data type of the property value, e.g., Text, Integer, Float, Date/Time. A property called “age” might be an Integer type, while “lastName” would most likely 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 “petPeeves” would most likely be multi-valued.
- 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: A default value provided by the property set corresponding to the property. A property called “favoriteDay” might have a default value of “Saturday.”

For Personalization Server purposes, property sets are applied to four 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. Please see the "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 Personalization 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 Personalization 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. Please see "Creating and Managing Content" for more complete information on this subject.

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 a statement that describes the purpose of the property set. Properties can be copied from an existing property set if a preexisting property set defines similar properties. Expanding the previous example, if portal A's properties have been defined and portal B is going to have the same (or similar) properties, then time is saved by copying the properties from portal A's property set when creating portal B'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
- Description
- Type
- Selection option
- Creation category

Name is the name of the property, such as “backgroundColor”. Description is a textual description of the property, perhaps describing the purpose of the property. Type is the data type of the property value. Date 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.

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 you – 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.axiom.contact.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

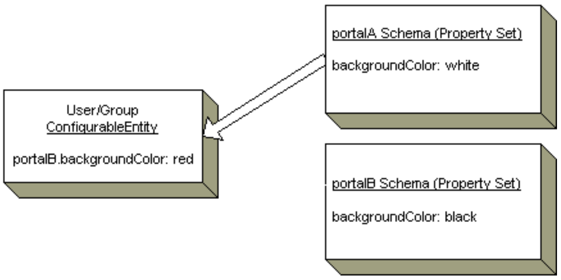

User/Group
ConfigurableEntity
portalB.backgroundColor: red
portalA Schema (Property Set)
backgroundColor: white
portalB Schema (Property Set)
backgroundColor: black

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.

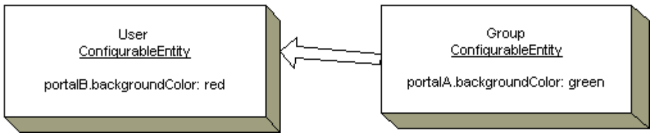
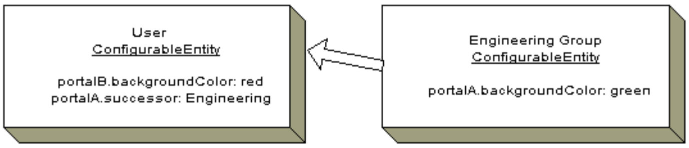


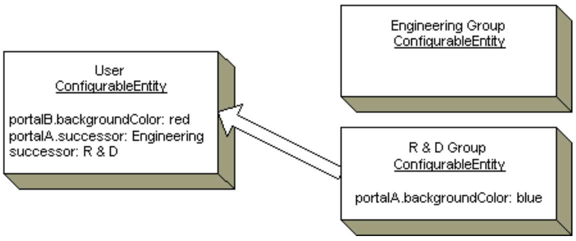
Figure 3-4 provides an example of an implicit successor. An implicit successor is a successor tied to a particular Property Set. In this case, the user does not have a value for portalA.backgroundColor, and no explicit successor is provided in the `getProperty()` call. However, the group has already 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.


User
ConfigurableEntity
portalB.backgroundColor: red
portalA.successor: Engineering
Engineering Group
ConfigurableEntity
portalA.backgroundColor: green

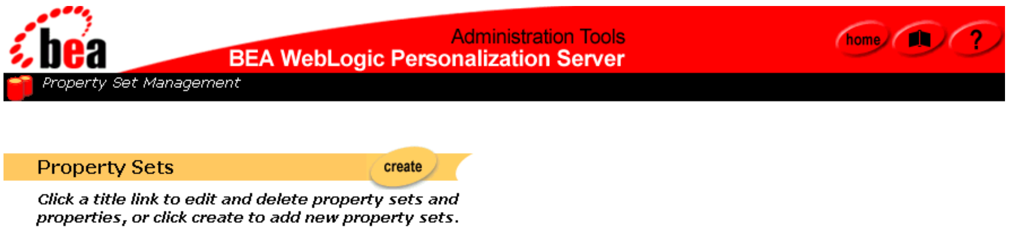
There also exists the notion of a default successor, which can be searched after an explicit successor and a Property Set-related successor have failed to return a value for the property. Figure 3-5 shows such a case. In this example, the Property Set-related successor cannot produce the necessary property 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” on page 4-3.

Using the Property Set Management Tool



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*.

   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.

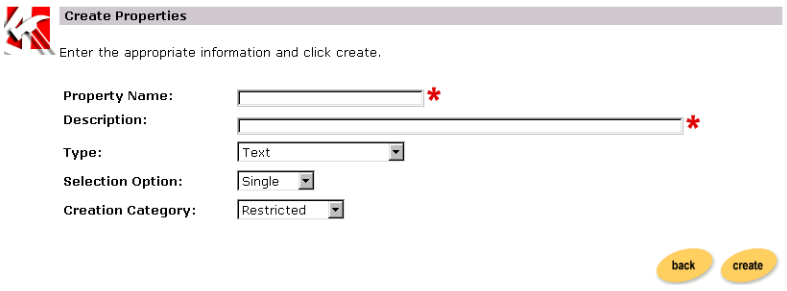
c. Select the appropriate property set containing the properties you want copied from the Copy Properties From drop-down list box.

3. Click **create** to create the property set or click **back** to return to the Property Set Management Home Page without saving the property set. The Property Set Management home page appears after the create operation completes.

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, select the appropriate title link for the property set you wish to add a property to. The Property Set view page appears.
3. Click **create** on the Properties bar. The Create Properties page appears.


Create Properties

Enter the appropriate information and click create.

Property Name:	*
Description:	*
Type:	Text
Selection Option:	Single
Creation Category:	Restricted

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 (e.g., Property: Color, Attribute: red). The multiple option refers to those properties having multiple options (e.g., Property: Colors, Attributes: red, green, blue, etc.).

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, etc. 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.

To set up the property default value:

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

For single/restricted:

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 check box(es).
6. Click **save**.

For single/unrestricted:

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**.

For multiple/restricted:

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 by selecting the appropriate radio button.
6. Click **save**.

For multiple-unrestricted:

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**.

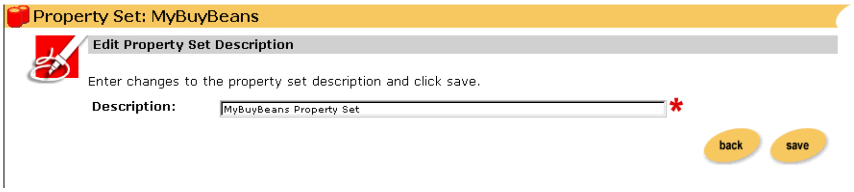
Editing Property Sets

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:

   a. Click **edit** on the Property Set Description bar. The Edit Property Set page appears.


Property Set: MyBuyBeans

Edit Property Set Description

Enter changes to the property set description and click save.

Description: MyBuyBeans Property Set *

b. Enter the new description in the Description field.

c. 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 value 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

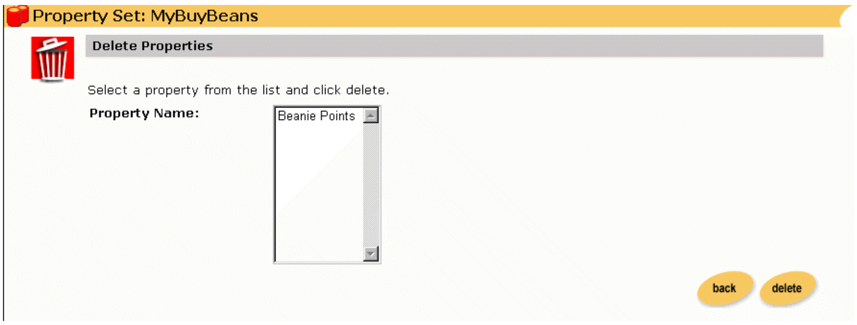
To delete a property set:

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.


Property Set: MyBuyBeans

Delete Properties

Select a property from the list and click delete.

Property Name: Beanie Points

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.

CHAPTER

4 Creating and Managing Users

The following topics are covered here:

Viewing LDAP configuration settings
Selecting groups for the Personalization Server from realm
Mapping realm groups to the Personalization Server
Deleting groups from your database

User Management joins enterprise data about users with profile data that is used to personalize the users' view of the application.

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 follow:

- **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. A user management system can either provide a default schema for user-group information persistence or interface with existing user databases via standardized interfaces (e.g., LDAP) or customized connectors.
- **User Profile Management**—To create an effective personalization server solution, a user profile must be made available to other personalization subsystems. 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 follows:

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

Group – `com.beasys.commerce.axiom.contact.Group`

ConfigurableEntity –
`com.beasys.commerce.axiom.foundation.ConfigurableEntity`

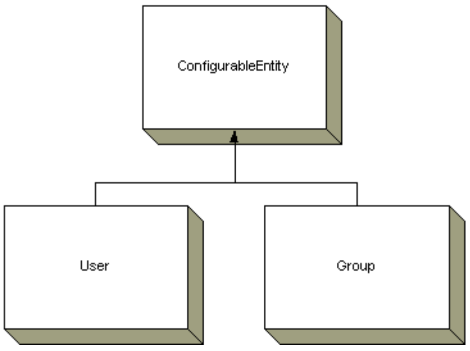
The User Management system works in conjunction with the WebLogic Server's security realm. In this arrangement, the security realm provides a list of users and groups, group membership information, and authentication. The User Management system uses the security realm to authenticate users and to know which users and groups exist and are valid, and what 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 doesn't 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 (e.g., 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. The figure below 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 email 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.

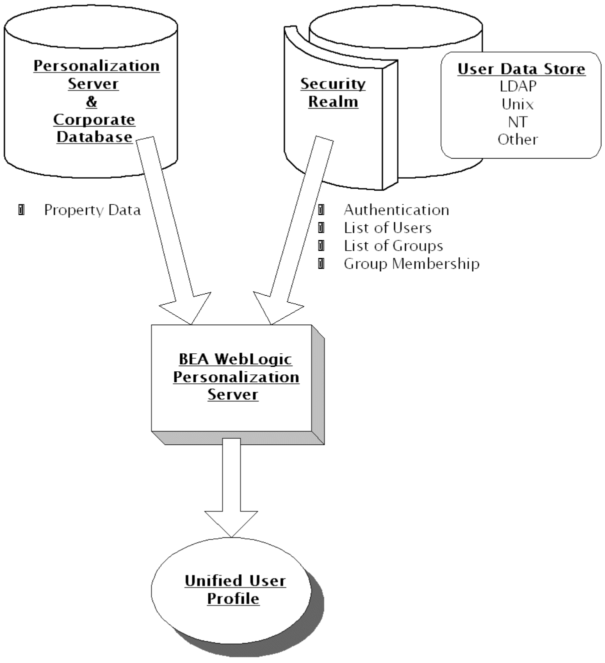
The Unified User Profile (UUP) is the name used to describe this aggregation of properties from an existing data source and the Personalization Server database tables into a single, customized user profile. More specifically, a UUP marries existing user/customer data by extending BEA's User component. 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.

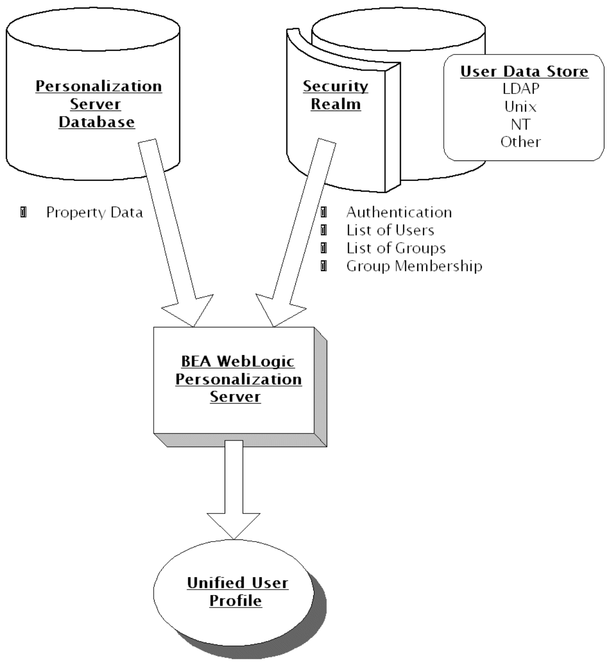
Figure 4-2 Configuration 1



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.

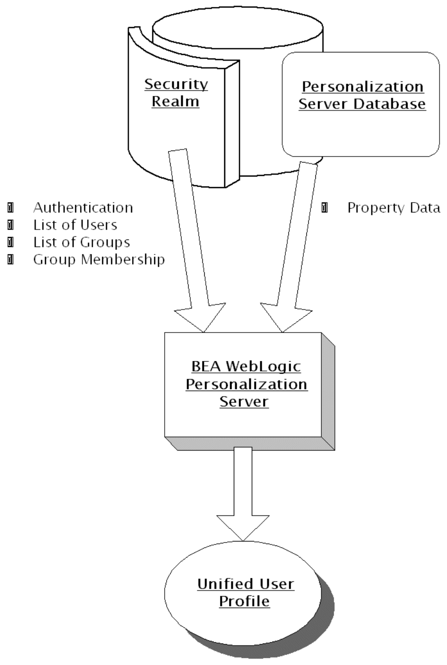
Figure 4-3 Configuration 2



Configuration 3

There is no existing store of users and groups. The Personalization Server's database tables contain all user and group data.

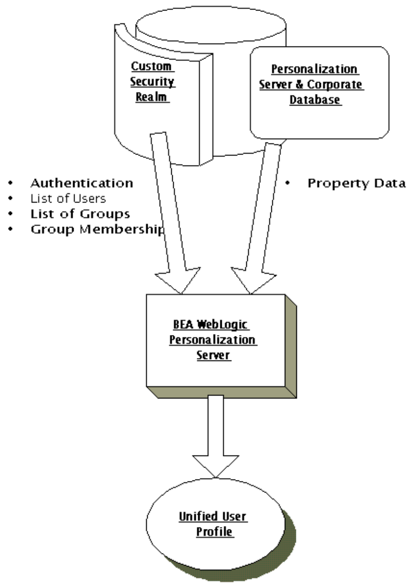
Figure 4-4 Possible Configuration 3



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.

Figure 4-5 Possible Configuration 4



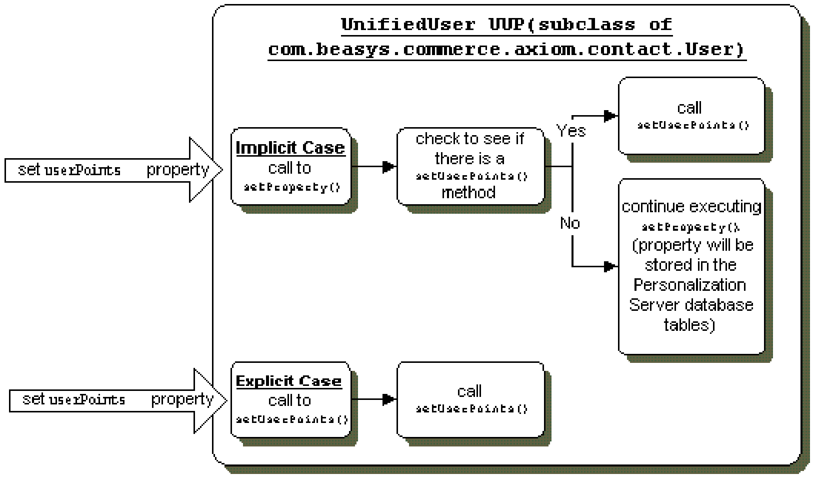
The UnifiedUser example, found at `<install_dir>/server/publish.html/examples/unifieduserprofile/index.html`, demonstrates a fictitious company's use of the UUP to take advantage of existing customer data. The UnifiedUser extends `com.beasys.commerce.axiom.contact.User` and retrieves data from a preexisting 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	`com.beasys.commerce.axiom.contact.User-Pk--with no key fields added.`
UUP EJB Interface	`com.beasys.commerce.axiom.contact.User`
UUP EJB Implementation	`com.beasys.commerce.axiom.contact.UserImpl`

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.

The figure below 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.


UnifiedUser UUP(subclass of com.beasys.commerce.axiom.contact.User)
set userPoints property
Implicit Case
call to setProperty()
check to see if there is a setUserPoints() method
Yes
No
call setUserPoints()
continue executing setProperty() (property will be stored in the Personalization Server database tables)
set userPoints property
Explicit Case
call to setUserPoints()
call setUserPoints()

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 in the entity for the property for the specified Property Set.
2. Look in the entity for the property in the default (null) Property Set.
3. Look in the entity for the property in the Reserved Property Set (for properties from LDAP if using the LDAPRealm).
4. Look for the property in the entity's explicit successor (if specified).
5. Look for the property in the entity's successor for the specified Property Set.
6. Look for the property in the entity's default successor.
7. Look for a default value as defined in the Property Set if the Property Set is specified (not null).
8. Return the defaultValue passed into the `getProperty()` method.

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

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

This method has a few details that it 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 doesn’t 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” or “default” property set are editable from the tools.

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

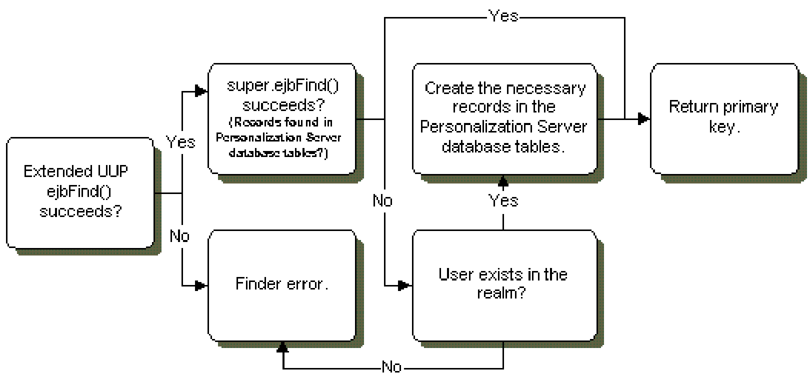
```
Object value     = myUser.getProperty(“my_property_set”,
                                      "my_integer_property",
                                       null,
                                       null);
                  Number tempNumber = (Number) value;
                  int    realValue  = 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 Flow during an `ejbFind()` operation.


Yes
Extended UUP ejbFind() succeeds?
Yes
No
super.ejbFind() succeeds? (Records found in Personalization Server database tables?)
Finder error.
No
Create the necessary records in the Personalization Server database tables.
Yes
User exists in the realm?
No
Return primary key.

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 collaborate 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-entr
y-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. SmartConnectionPoolClass

This environment entry is used to configure the EJB to use the correct database connection pool class. The value must always be:

com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool

Exact Entry:

```
<env-entry>
<env-entry-name>SmartConnectionPoolClass</env-entry-name>
<env-entry-type>java.lang.String</env-entry-type>
<env-entryvalue>
```

```
com.beasys.commerce.foundation.plugin.weblogic.
WeblogicConnectionPool
```

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

4. SmartBMP_URL

This environment entry provides the URL of the database to the EJB. The value must always be:

jdbc:weblogic:jts:commercePool

Exact Entry:

```
<env-entry>
<env-entry-name>SmartBMP_URL</env-entry-name>
<env-entry-type>java.lang.String</env-entry-type>
<env-entry-value>jdbc:weblogic:jts:commercePool</env-entry-value>
</env-entry>
```

5. 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>
```

6. EntityPropertyManager

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.EntityPropertyAggre
gator</env-entry-value>
</env-entry>
```

For any other configuration the `EntityPropertyManager` 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.

```
<ejb-jar>
   <enterprise-beans>
     <entity>
     <ejb-name>com.beasys.commerce.user.example.UnifiedUser</ejb-name>
     <home>com.beasys.commerce.user.example.UnifiedUserHome</home>
     <remote>com.beasys.commerce.user.example.UnifiedUser</remote>
     <ejb-class>com.beasys.commerce.user.example.UnifiedUserBean</ejb-class>
     <persistence-type>Bean</persistence-type>
     <prim-key-class>com.beasys.commerce.user.example.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>SmartConnectionPoolClass</env-entry-name>
         <env-entry-type>java.lang.String</env-entry-type>
         <env-entry-
value>com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool</en
v-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>SmartBMP_URL</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>jdbc:weblogic:jts:commercePool</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>
         </entity>
        </enterprise-beans>
   </ejb-jar>
```

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:

Profile Type Name	**Arbitrary name that is later used to refer to the profile type through the User Management system's** `<um:getprofile>` **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.OracleDriv
er
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 AclEntries, 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 for doing so.

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 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`*`”>`. See the documentation for User Management JSP tags for more information on these tags.

For an example, see <install_dir>/server/public_html/anonymousprofile/index.html.

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

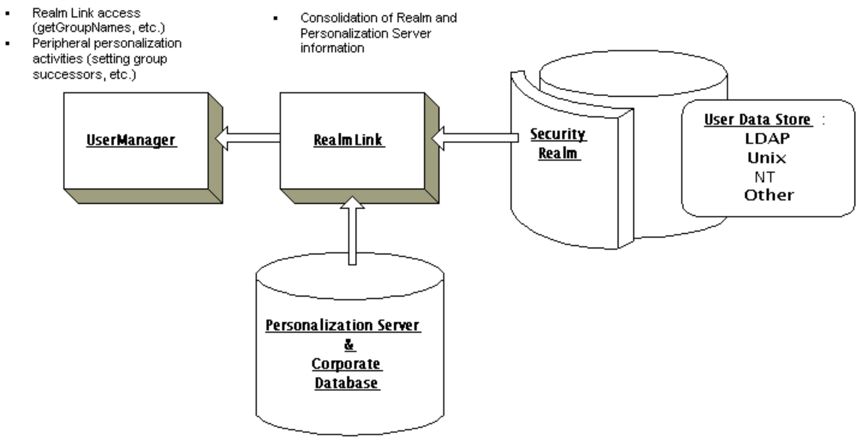
- Adding groups to groups/Removing groups from groups
- Retrieving usernames corresponding to a group
- Retrieving group names corresponding to a user
- Retrieving unique group and user IDs based on group/user name
- Retrieving group/user name based on unique ID
- Retrieving user/group objects based 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.

The figure below 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 UserManager/RealmLink Cooperation.


Realm Link access (getGroupNames, etc.)
Peripheral personalization activities (setting group successors, etc.)
Consolidation of Realm and Personalization Server information
UserManager
RealmLink
Security Realm
User Data Store :
LDAP
Unix
NT
Other
Personalization Server & Corporate Database

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

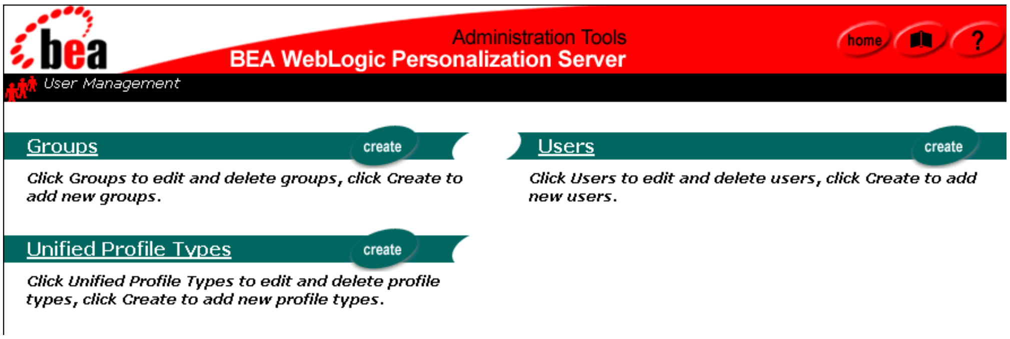
1. For certain operations, (`com.beasys.commerce.axiom.contact.UserManager`), the UserManager qualifies the security realm being used before taking action. These operations can only be performed if the current security realm class is `com.beasys.commerce.axiom.contact.security.RDBMSRealm`. See UserManager 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 (e.g.,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



Click Groups to edit and delete groups, click Create to add new groups.

Click Users to edit and delete users, click Create to add new users.

Click Unified Profile Types to edit and delete profile types, click Create to add new profile types.

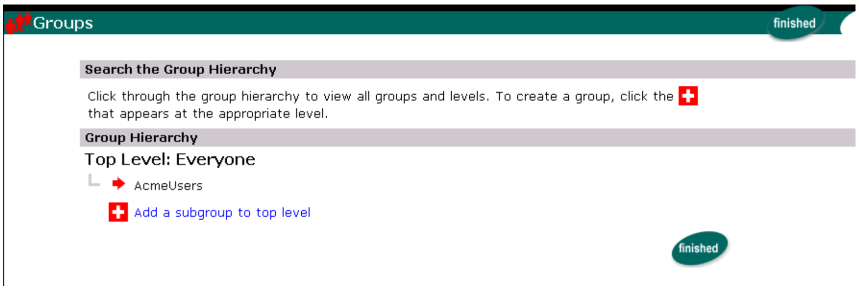
The User Management tools allow you to create and associate users and groups or to link to and use existing directories of users. A user or group may then be personalized by overriding property values as defined in the Property Set Management tool. The Unified Profile Types tool allows you to configure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

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.



Search the Group Hierarchy

Click through the group hierarchy to view all groups and levels. To create a group, click the ✚ that appears at the appropriate level.

Group Hierarchy

Top Level: Everyone

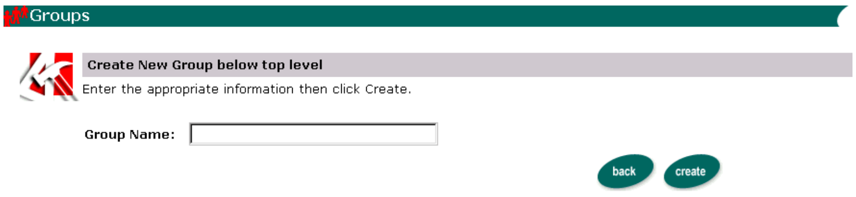
AcmeUsers

Add a subgroup to top level

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.



Create New Group below top level

Enter the appropriate information then click Create.

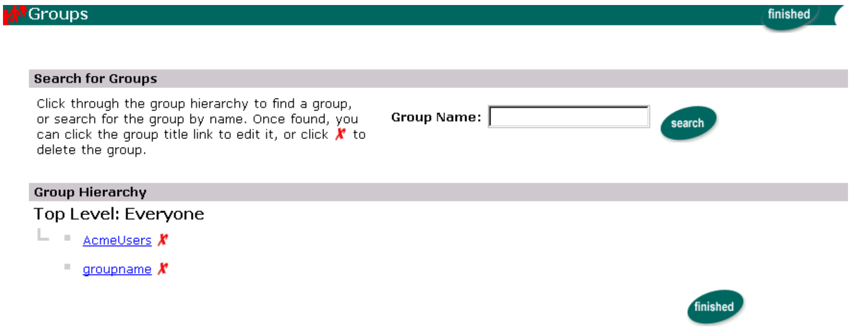
Group Name:

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

Deleting 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.



Search for Groups

Click through the group hierarchy to find a group, or search for the group by name. Once found, you can click the group title link to edit it, or click ✗ to delete the group.

Group Name:

Group Hierarchy

Top Level: Everyone

AcmeUsers ✗

groupname ✗

To locate the group to delete by name:

a. Enter the group name in the Group Name field.

Note: The group name must be entered exactly.

b. Click **search**.

or

To locate the group to delete within the Group Hierarchy:

- Navigate the Group Hierarchy tree view.

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

Adding 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 by name:

   a. Enter the group name in the Group Name field.

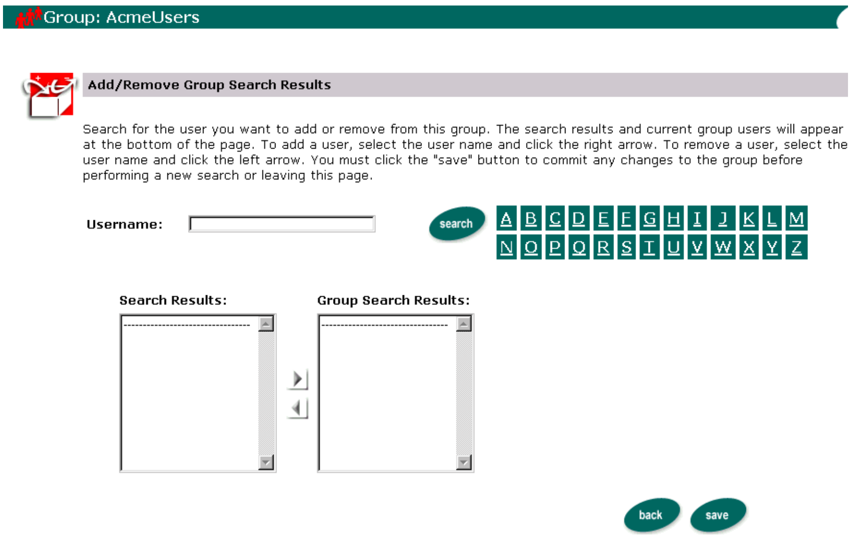
b. Click **search**.

or

To locate the appropriate 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:

To locate the user by name:

a. Enter the user name in the Username field or a partial user name, including an asterisk.

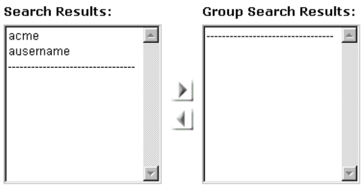
b. Click **search**. The Search Results and the current Group Users appear at the bottom of the page.

or

To see a list of all users in an alphabetized category:

- Click the appropriate letter corresponding to the first letter of the user name. The Search Results and the current matching Group Users appear at the bottom of the page.

5. Select the user name, or a group of names, from the Search Results field.



6. Click the left-to-right directional arrow. The user name(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

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 by name:

   a. Enter the group name in the Group Name field.

   b. Click **search**.

   or

   To locate the appropriate 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 the user by name:

a. Enter the user name or a partial user name, including an asterisk, in the Username field.

b. Click **search**. The Search Results and the current matching Group Users 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 user name. The Search Results and the current Group Users appear at the bottom of the page.

5. Select the user name, or a group of user names, from the Group Users field.
6. Click the right-to-left directional arrow. The user name(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

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 by name:

   a. Enter the group name in the Group Name field.

   b. Click **search**.

   or

   To locate the appropriate group within the Group Hierarchy:

   - Navigate the Group Hierarchy tree view.
3. Select the group. The Group Properties view appears.

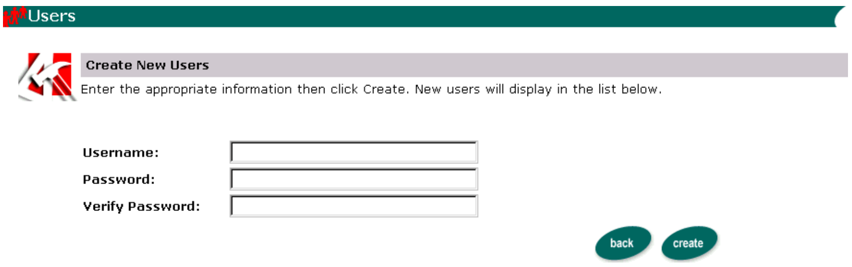
4. Select or search for a property set to view for this group. For specific instructions on property set management, see **Creating and Managing Property Sets**. The group's default property values appear if no other property set has been accessed during the tools session.
5. Click **search**.
6. Click **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.

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

Creating users

Note: All new users are com.beasys.commerce.axiom.contact.usercomponents.

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.



Create New Users

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

Username:

Password:

Verify Password:

3. Enter the user name in the Username field.

Note: Limit user names to 25 characters.

4. Enter the password associated with the User Name 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. 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 RDBMS realm allows mixed case (e.g., *User*, *user*) user creation.

Editing user property values

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

Note: Explicit properties of UUP are not editable from the administration tools.

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.

Note: Use the wildcard feature by entering a partial user name immediately followed by an asterisk (*). The asterisk is a search return variable.

To locate the appropriate user by name:

a. Enter the user name in the Username field.

b. Click **search**. The search results appear at the bottom of the page.

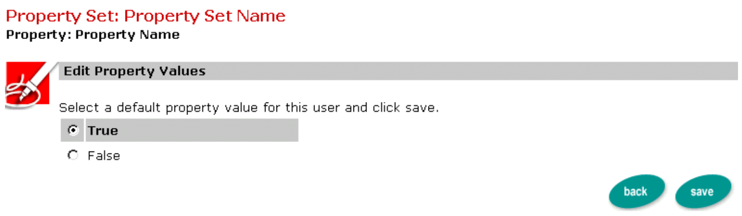
or

To see a list of all users within an alphabetized category:

- Click the appropriate letter corresponding to the first letter of the user name. A list of users appear at the bottom of the page.

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


Property Set: Property Set Name
Property: Property Name

Edit Property Values

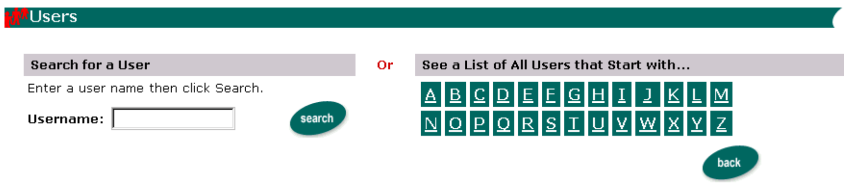
Select a default property value for this user and click save.

[✓ True]
False

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.

Deleting 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.


Users

Search for a User

Enter a user name then click Search.

Username:

Or

See a List of All Users that Start with...

To locate the appropriate user by name:

a. Enter the user name in the Username field or a partial name with an asterisk.

b. Click **search**. The search results appear at the bottom of the page.

or

To see a list of all users within an alphabetized category:

- Click the appropriate letter corresponding to the first letter of the user name. A list of users appear at the bottom of the page.

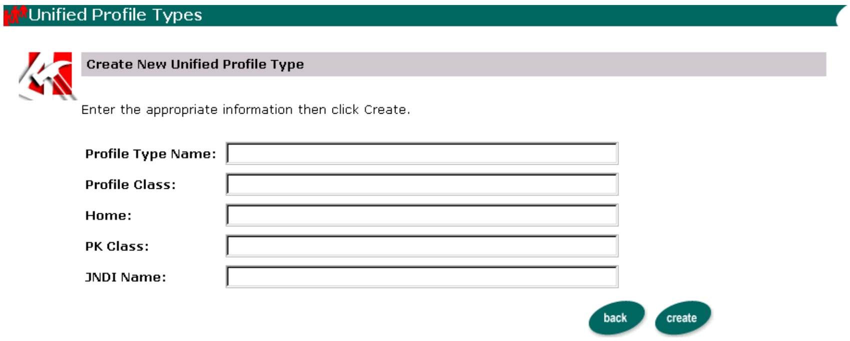
3. Click the **X** to right of the user name to delete the user. A confirmation dialog box appears.
4. Click **OK** to confirm the deletion.

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 user name will continue to be returned from user searches.

Creating 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.



Enter the appropriate information then click Create.

Profile Type Name:	
Profile Class:	
Home:	
PK Class:	
JNDI Name:	

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 <um:getprofile> 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

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

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 user name to delete the user. A confirmation dialog box appears.
4. Click **OK** to confirm the deletion.

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.

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/docs50/admindocs/ldap.html for details.

Registering group attributes for retrieval from LDAP

This screen is used to register group attribute names for runtime retrieves via the group profile.

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. 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**.

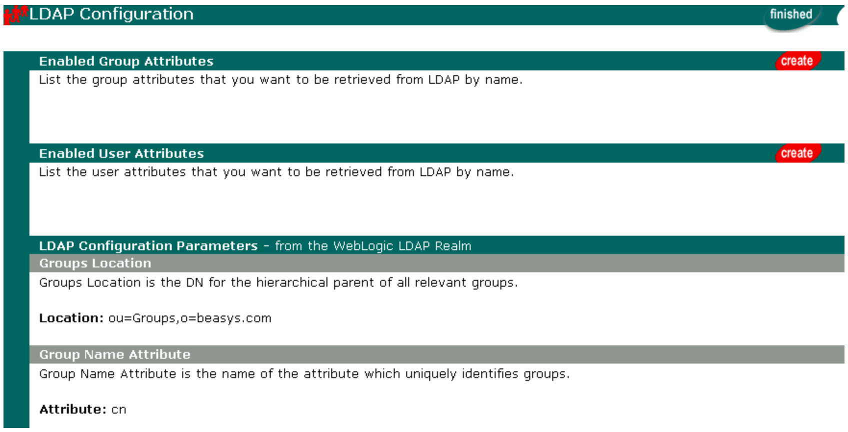
Deleting user attributes from LDAP

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**.

Adding group attributes in 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.



Enabled Group Attributes
List the group attributes that you want to be retrieved from LDAP by name.

Enabled User Attributes
List the user attributes that you want to be retrieved from LDAP by name.

LDAP Configuration Parameters - from the WebLogic LDAP Realm

Groups Location
Groups Location is the DN for the hierarchical parent of all relevant groups.

Location: ou=Groups,o=beasys.com

Group Name Attribute
Group Name Attribute is the name of the attribute which uniquely identifies groups.

Attribute: cn

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/docs50/admindocs/ldap.html for details.

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 in 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

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 following parameters from the LDAP Configuration Parameters field:

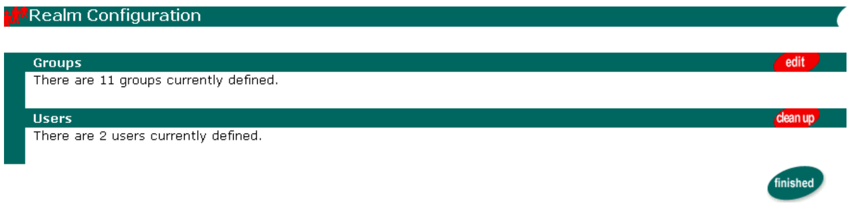
Parameter	Description
Groups Location	Distinguished name for the hierarchical parent of all relevant groups.
Group Name Attribute	The name of the attribute that uniquely identifies a group.
Group Username Attribute	The name of the attribute in group objects that has as its value the group members.
Users 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.

Parameter	Description
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 the Personalization Server from realm

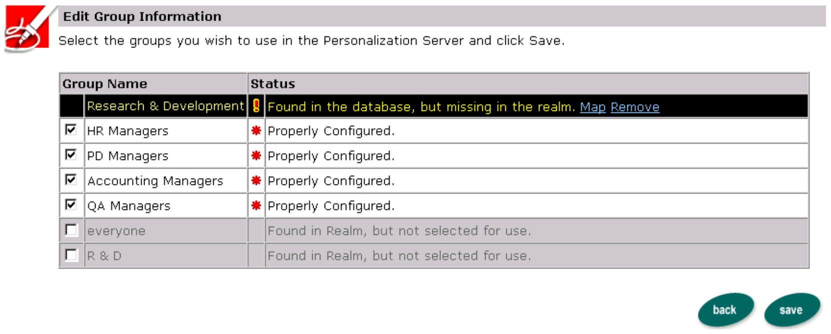
1. On the Administration Tools Home Page, click the User Management icon. The User Management home page appears.
2. On the User Management home page, click **Realm** in the Realm banner. The Realm Configuration page appears.



There are 11 groups currently defined.

There are 2 users currently defined.

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



Edit Group Information

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

	Group Name		Status
	Research & Development	!	Found in the database, but missing in the realm. Map Remove
☑	HR Managers	✱	Properly Configured.
☑	PD Managers	✱	Properly Configured.
☑	Accounting Managers	✱	Properly Configured.
☑	QA Managers	✱	Properly Configured.
☐	everyone		Found in Realm, but not selected for use.
☐	R & D		Found in Realm, but not selected for use.

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

Mapping realm groups to the Personalization Server

Use this tool when a group name changes in the realm. The records in the Personalization Server to reflect the 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.

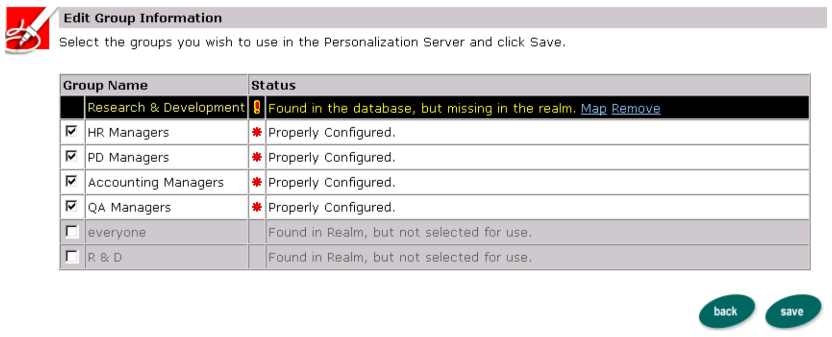
Note: You are only given the option of mapping those groups that have been found in your database but are missing from the realm.

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

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.



Edit Group Information

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

	Group Name		Status
	Research & Development	!	Found in the database, but missing in the realm. Map Remove
☑	HR Managers	✱	Properly Configured.
☑	PD Managers	✱	Properly Configured.
☑	Accounting Managers	✱	Properly Configured.
☑	QA Managers	✱	Properly Configured.
☐	everyone		Found in Realm, but not selected for use.
☐	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 from 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 user names are no longer needed. All associated records are removed.

CHAPTER

5 Creating and Managing Content

Creating and Managing Content covers the following topics:

The Content Management component of WLPS 2.0 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.

What is the Content Management Component?

The Content Management runtime component 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.

WebLogic Personalization Server 2.0 provides several components that allow content personalization for users. Together, these components provide a complete personalization solution. Of these personalization components, the Portal, Rules, User, and Property Set Management elements include edit-time graphical user interfaces (GUIs) that allow developers to customize the elements. Neither the Content Management or Personalization Advisor components have a GUI.

The Content Management component works alongside the other components to deliver personalized content, but doesn't 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` (i.e. the `getIdentifier` method).
  - *mimeType*: Corresponds to the `String` MIME type of the `Content` (i.e. 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 (i.e. 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 (i.e. the `getVersion` method).
  - *author*: Corresponds to the `String` identifier of the author of the document (i.e. the `getAuthor` method).
  - *creationDate*: Corresponds to the `Timestamp` of when the document was created (i.e. the `getTimestamp` method).
  - *modifiedBy*: Corresponds to the `String` identifier of the individual who last modified the document (i.e. the `getModifiedBy` method).
  - *modifiedDate*: Corresponds to the `Timestamp` of when the document was last modified (i.e. the `getModifiedDate` method).
  - *lockedBy*: Corresponds to the `String` identifier of the individual who has the document locked (i.e. the `getLockedBy` method).
  - *description*: Corresponds to the `String` description of the document (i.e. the `getDescription` method).
  - *comments*: Corresponds to any `String` comments about the document (i.e. 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 5-15 for more information about queries.

Third-party tools and WLPS

BEA partners with third-party vendors to add flexibility to WebLogic Personalization Server. The Content Management component works with Interwoven's TeamSite/OpenDeploy product and Documentum's 4i product. Both these products provide robust content creation management solutions while the Content Management component of WLPS 2.0 personalizes and serves the content to the end-user.

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 //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 2.0 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**
contentHome	maybe	If the *`contentHome`* initialization parameter is not specified, then this is required and will be used as the JNDI name of the DocumentHome. If the *`contentHome`* initialization parameter is specified, this is ignored.
contentId	no	The string identifier of the Document to retrieve. If not specified, the servlet looks in the `PATH_INFO`.
blockSize	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, the *`Content-Length`* to the `Document’s` size, and correctly sets the *`Content-Disposition`*, which should present the correct file name when the file is saved from a browser.

Example 1: Usage in a JSP:

```
<cm:select contentHome="bea.eDocs.CMgr" 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=bea.eDocs.CMgr"><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=”bea.eDocs.cMgr” 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><img src=”/showDocServlet?contentId=<cm:printproperty
        id=”img” name=”identifier”
        encode=”url”/>&contentHome=”bea.eDocs.cMgr”>
    </es:foreachinarray>
</ul>
```

JSP Tags

The Content Management component includes 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 tags include:

- The `<cm:select>` tag uses only the search expression query syntax to select content. See the JSP documentation for more information.
- The `<cm:selectbyid>` tag retrieves content using the content's unique identifier. See the JSP documentation for more information.
- The `<cm:printproperty>` tag inlines the value of the specified Content metadata property as a string. See the JSP documentation for more information.
- The `<cm:printdoc>` tag inlines the raw bytes of a Document object into the JSP output stream. See the JSP documentation for more information.

Configuring the Content Management component

The Document EJB, Document Schema 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 WLS 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 EJB deployment descriptor

The logic for loading Document EJBs is handled via a `SmartBMP`. The Document 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 class must be capable of populating a `DocumentImpl` object and must also have the methods defined in the `Content` and `Document` Javadocs.

To use the reference implementation document management system, set `SmartBMPClass` to `com.beasys.commerce.axiom.document.SPIDocumentSmartBMP` and specify the following EJB environment variables in the document EJB deployment descriptor:

- *`SmartConnectionPoolClass`* (required): Specifies the fully-qualified class name of the `SmartConnectionPool` implementation class. In WebLogic Server, set *`SmartConnectionPool`* to `com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool`.
- *`SmartBMPUpdate`*: Set to *`false`*.
- *`SmartBMP_URL`* (required): Specifies the *`JDBC URL`* to the document JDBC connection pool (see "Setting up Connection pools" on page 5-12), which is the URL that the EJB uses to obtain a document connection. This value should correspond to the WebLogic connection pool that uses the document reference implementation JDBC driver.
- *`PropertyCase`*: This sets how the `DocumentImpl` modifies incoming property names. If this is *lower*, all property names are converted to 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 (e.g. the Documentum `SmartBMP` expects everything in lower case). For the document reference implementation, do not specify the *`PropertyCase`*.

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

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:

- *SmartConnectionPoolClass* (required): Specifies the fully-qualified class name of the `SmartConnectionPool` implementation class. In WebLogic Server, set *SmartConnectionPool* to `com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool`.
- *SmartBMPUpdate*: Set to *false*.
- *SmartBMP_URL* (required): Specifies the JDBC URL to the document JDBC connection pool (see "Setting up Connection pools" on page 5-12), which is the URL that the EJB uses to obtain a document connection. This value should correspond to the WebLogic connection pool that uses the document reference implementation JDBC driver.

  Note: This value should correspond to the value in the Document EJB. See "Configuring the Document EJB deployment descriptor" on page 5-7 for more information.

Other `SmartBMP` class for other document management system 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:

- *UseDefaultHomeNames*: If this set to *true*, then the default home names will be used if either `ContentHome` or `SchemaHome` is not specified.
- *ContentHome*: This specifies the JNDI home name of the `DocumentHome` object to use.
- *SchemaHome*: This specifies the JNDI home name of the `SchemaHome` object to use.

Example deployment descriptor file

The following is a sample `ejb-jar.xml` deployment descriptor file:

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
1.1//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd'>
<ejb-jar>
  <enterprise-beans>

    <!-- our Document entity bean -->
    <entity>
      <ejb-name>com.beasys.commerce.axiom.document.Document</ejb-name>
      <home>com.beasys.commerce.axiom.document.DocumentHome</home>
      <remote>com.beasys.commerce.axiom.document.Document</remote>
      <ejb-class>com.beasys.commerce.axiom.document.DocumentImpl</ejb-class>
      <persistence-type>Bean</persistence-type>
      <prim-key-class>
          com.beasys.commerce.axiom.document.DocumentPk
      </prim-key-class>
      <reentrant>False</reentrant>
      <env-entry>
        <env-entry-name>SmartConnectionPoolClass</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>
           com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool
        </env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>SmartBMP_URL</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>jdbc:weblogic:pool:docPool</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.document.SPIDocumentSmartBMP
        </env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>SmartBMPUpdate</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>false</env-entry-value>
      </env-entry>
    </entity>

    <!-- our Schema entity bean -->
    <entity>
    <ejb-name>com.beasys.commerce.axiom.document.DocumentSchema</ejb-name>
    <home>com.beasys.commerce.foundation.property.SchemaHome</home>
    <remote>com.beasys.commerce.foundation.property.Schema</remote>
    <ejb-class>com.beasys.commerce.foundation.property.SchemaImpl</ejb-class>
    <persistence-type>Bean</persistence-type>
    <prim-key-class>
       com.beasys.commerce.foundation.property.SchemaPk
    </prim-key-class>
    <reentrant>False</reentrant>
    <env-entry>
      <env-entry-name>SmartConnectionPoolClass</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>
         com.beasys.commerce.foundation.plugin.weblogic.WeblogicConnectionPool
      </env-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>SmartBMP_URL</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>jdbc:weblogic:pool:docPool</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.document.SPISchemaSmartBMP
      </env-entry-value>
    </env-entry>
    <env-entry>
      <env-entry-name>SmartBMPUpdate</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>false</env-entry-value>
    </env-entry>
  </entity>

  <!-- The default DocumentManager bean -->
  <session>
    <ejb-name>com.beasys.commerce.axiom.document.DocumentManager</ejb-name>
    <home>com.beasys.commerce.axiom.document.DocumentManagerHome</home>
    <remote>com.beasys.commerce.axiom.document.DocumentManager</remote>
    <ejb-class>
        com.beasys.commerce.axiom.document.DocumentManagerImpl
    </ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>
    <env-entry>
      <env-entry-name>ContentHome</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>com.beasys.commerce.axiom.document.Document
```

```
        </env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>SchemaHome</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>
         com.beasys.commerce.axiom.document.DocumentSchema</env-entry-value>
      </env-entry>
    </session>
  </enterprise-beans>

  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>com.beasys.commerce.axiom.document.Document</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>*</method-name>
      </method>

      <method>
        <ejb-name>com.beasys.commerce.axiom.document.DocumentSchema</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>*</method-name>
      </method>

      <method>
        <ejb-name>com.beasys.commerce.axiom.document.DocumentManager</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>*</method-name>
      </method>

      <trans-attribute>Supports</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

Setting up Connection pools

For the document reference implementation, set up a specialized WebLogic connection pool with the same name as the `Document` and `Schema` EJB's *`SmartBMP_URL`* environment variable (see "Configuring the Document EJB deployment descriptor" on page 5-7).

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

- the *`SmartBMP_URL`* environment variable should be `jdbc:weblogic:pool: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 the 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.

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.Re
fDocumentProvider,\
    driver=com.beasys.commerce.axiom.document.jdbc.Driver,\
    loginDelaySecs=1,\
    initialCapacity=1,\
    maxCapacity=5,\
    capacityIncrement=1,\
    allowShrinking=true,\
    shrinkPeriodMins=15,\
    refreshMinutes=10,\
    props=jdbc.url=jdbc:weblogic:pool:commercePool;\
    jdbc.isPooled=true;\
    docBase=C:/WeblogicCommerce/docBase;\
    schemaXML=C:/WeblogicCommerce/docSchemas;\
    iw.schemaBase=C:/iw-home/templatedata
```

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

- JSP tags (see "Using Content Management JSP Tags" on page 5-22.)
- ContentHelper (see the API documentation)
- ContentManager (see the API documentation)
- ContentHome (see the API documentation)

Structuring a query

WLPS 2.0 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:

Note: Consult the API documentation on `com.beasys.commerce.util.ExpressionHelper` for more information about the query syntax.

```
attribute_name comparison_operator literal_value
```

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 doesn't comply with the Java variable-naming convention that uses alphanumeric characters.
  - `toProperty ('My Property') = 'Content'`
- Use \ along with the appropriate character(s) to create an escape sequence that include special characters in string literals.
  - `toProperty ('My Property\'s Contents') = 'Content'`
- 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 (e.g. *`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 table summarizes the comparison operators available for each metadata type. (See Support for Native Types in the Developer's Guide topic Overview of Personalization Development for more information about the native types supported in WLPS 2.0.)

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 `java.lang.Number`.
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 (i.e. the *`like`* 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 (e.g. 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 `java.sql.Timestamp`.
Multi-valued Comparison Operators (contains, containsall)	Multi-valued attributes support a *`contains`* 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 *`containsall`* operator, which takes another collection of objects of the attribute's subtype and checks that the attribute's value contains all of them. 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 true. For example, if the multi-valued text attribute *`keywords`* has the values *BEA*, *Computer*, and *WebLogic* and the operand is *BEA*, then the < operator returns true (*BEA* is less than *Computer*), the > operator returns false (*BEA* is not greater than any of the values), and the == operator returns true (*BEA* is equal to *BEA*).
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 (!).

Using the BulkLoader to load file-based content

WebLogic Personalization Server 2.0 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.

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]
  [-properties <name>] -conPool <name> [-schema <name>] [+schema]
  [-match <pattern>] [-ignore <pattern>] [-htmlPat <pattern>]
  [-d <dir>] [-mdext <ext>] [--] [files... directories...]
```

```
-verbose: emit verbose messages
+verbose: run quietly [default]
-recurse: recurse into directories [default]
+recurse: don’t recurse into directories
-delete: remove document from database
+delete: insert documents into database [default]
-metaparse: parse HTML files for <meta> tags [default]
+metaparse: don’t parse HTML files for <meta> tags
-cleanup: if specified, this only performs a table cleanup using the -d
    argument as the document base (i.e. all files will need to be under
    that directory).
+cleanup: turn off table cleanup (i.e. do a document load) [default]
-hidden: specify to ignore hidden files and directories [default]
```

```
+hidden: specify to include hidden files and directories
-inheritProps: specify to have metadata properties be inherited when
    recursing [default]
+inheritProps: specify to have metadata properties not be inherited
    when recursing.
-htmlPat <pattern>: Specifies a pattern for determining which files are HTML
    files for determining whether to do the <meta> tag parse. This can be
    specified mulitple 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.properites" 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, than no schema file will be created.
-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 starts with a ".". This defaults to ".md.properties".
--: 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 (e.g. *[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 file name 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 file name 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` be added to the properties from the parent directory. 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`. 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 (and possibly parent directories') properties.

Finally, 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. 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 any <meta> tags that contain name and content values found anywhere in the file (not just in the HTML head section) into the file's properties.

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 files's `.md.properties` file
4. If the file is an HTML file, then it uses `<meta>` tags.

The metadata is gathered in a last-seen-is-used algorithm. Therefore, for example, if a metadata attribute is specified in both the `<meta>` tags and the directory's dir.md.properties file, the value from the `<meta>` tags will be used.

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's when the document is being modified in the database).

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.

Cleaning up the database

If the `-cleanup` option is specified, the BulkLoader will not actually load any documents. Instead, it will attempt to cleanup 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.

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.

CHAPTER

6 Creating and Managing Rules

Creating and Managing Rules covers the following topics:

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.

What is the Rules Manager?

WebLogic Personalization Server 2.0 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 WLPS 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 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 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 instance, `Content.size < 10000` would specify a part of a query to find content items who have 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, 2000, 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 (see the Foundation documentation).
- `Session`: This object is used in the same way as the `User` object. The `Session` properties are defined in a default property set (see the Foundation documentation).

What are Rulesheets?

The BEA WebLogic Personalization Server 2.0 provides rulesheets that comprise a set of classifier and content selector rules. These rulesheets act as containers for rules that match personalized content with users.

A saved rulesheet generally contains a set of related rules and always has a relationship with a property set (see "Creating and Managing Property Sets" on page 3-1) that defines the attributes available for user and group profiles.

Note: Once you define a rulesheet and the property set to which it relates, you cannot change the relationship. You must create a new rulesheet and relate it to another property set if you want to change the property set for a rulesheet. This relationship between rulesheets and property sets makes it important to be careful when modifying property sets that rulesheets depend on. In general, you shouldn't change or delete properties if a rule refers to it. Adding properties does not affect existing rules.

Classifier rules

Classifier rules categorize users into groups using simple Boolean logic that 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 doesn't 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 (see "Content selector rules" on page 6-4) can use classifier rules as they select personalized content to match a user or group profile.

Use the <pz:div> (see the JSP documentation) JSP tag to include a classifier rule in a JSP page.

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 run-time 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 that 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*: 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

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, 2000, 00:00:00 MST"
    and Now < " Feb 01, 2000, 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
```

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

Use the `<pz:contentselector>` JSP tag to include content selector rules in JSP pages. (See the JSP documentation.)

Debugging rulesheets

You might notice that a rulesheet you've used in the past begins functioning oddly. This bug is probably because of a change in the property set with which the rulesheet has a relationship.

What is the relationship between Property sets and rulesheets?

Rulesheets rely on property sets to provide the properties they use to evaluate user and group profiles. If a property is modified after a rulesheet 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 rulesheet acts upon, any change to the properties the rules use will affect the schema and may alter the validity of the rulesheet. In general, be careful when modifying or deleting existing properties.

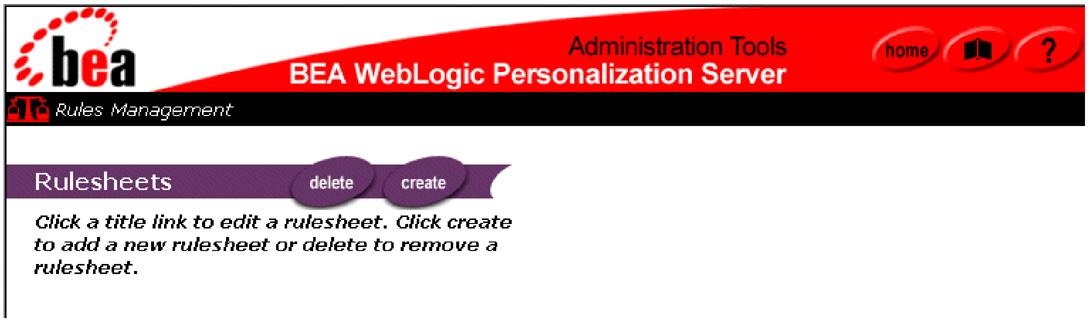
Note: You can add properties without affecting existing rules.

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 rulesheets.



Click a title link to edit a rulesheet. Click create to add a new rulesheet or delete to remove a rulesheet.

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

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

Creating a rulesheet

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 Rulesheet edit page appears.
3. Enter a name for the rulesheet in the Name field.

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

4. Enter a description for the rulesheet in the Description field.
5. Select a property set for the rulesheet from the choices in the Property Set drop-down list box.

Note: You cannot change the property set after you create the rulesheet because the rules in the rulesheet fit only the schema defined in the property set.

6. Click **create** to save the rulesheet with the parameters you've selected. WebLogic Personalization Server saves the new rulesheet 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 rulesheet.

Opening a rulesheet

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 rulesheet to open. The Rulesheet view page appears, displaying rulesheet information and a list of the classifier and content selector rules included in the rulesheet.

Editing rulesheet properties

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet you want to edit.
3. On the Rulesheet view page, click **edit** in the Definition bar. The Edit Rulesheet edit page appears.
4. Enter a new name and description.

Note: Remember you can’t change the property set. You must create a new rulesheet to change the property set. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet.

5. Click **save** to commit the changes or **back** to cancel the changes. If you save the changes, the Rulesheet view page appears with the new information displayed.

Saving a rulesheet

1. On the Rulesheet view page, click **finished**. The rulesheet 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 rulesheet is saved, so ensure you click **finished** before closing the rulesheet.

Deleting a rulesheet

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 rulesheet to delete from the Rulesheet Name drop-down list box.
4. Click **delete** to permanently delete the rulesheet 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 that appears to confirm the delete operation.

Navigating between rule types

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open a rulesheet. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet 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

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet which contains the rule you want to locate. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Locate the top of the rule type list (classifier or content selector type). See “Navigating between rule types” on page 6-9 for the procedure to navigate between rule types.
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.

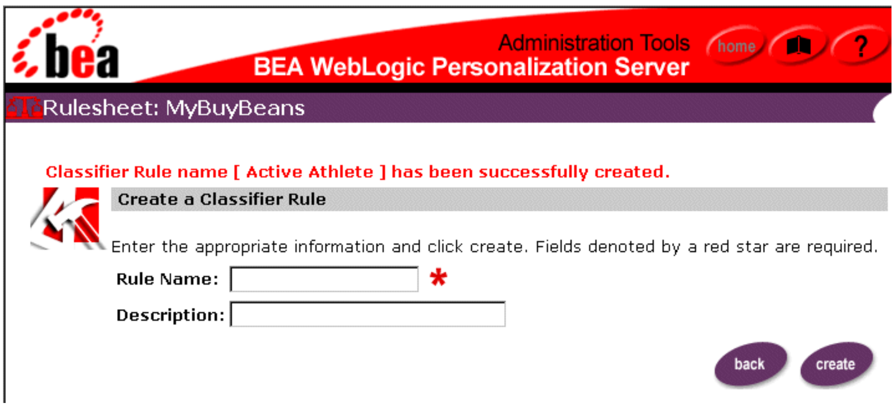
5. Locate the rule you need. See "Editing a rule" on page 6-11 for information about making changes to the rule.

Creating a classifier rule

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet in which you want to create the rule by clicking the rulesheet name. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet 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 rulesheet. 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 Rulesheet view page. The Rulesheet view page appears with the new rule(s) you created displayed in the alphabetical rule list.



Rulesheet: 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:

Editing a rule

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet that contains the rule by clicking the rulesheet name. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Locate the rule you want to edit. See "Finding a rule" on page 6-9 for information on locating a rule.
4. Click the name of the rule. The Rule view page appears.
5. Edit the rule using the instructions included in this documentation.

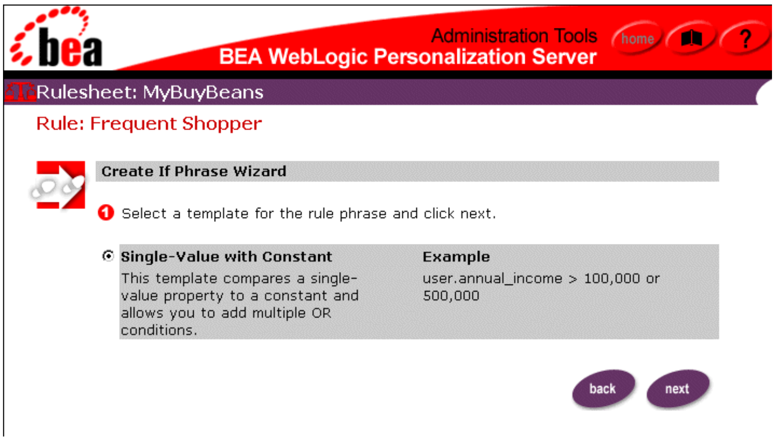
Editing rule properties

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet containing the rule you want to edit (see "Opening a rulesheet" on page 6-7). The Rulesheet view page appears.

3. Find the appropriate rule to edit. See “Finding a rule” on page 6-9 for information on locating a rule.
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

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet that contains the rule to which you want to add a phrase. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule you want to modify. See “Finding a rule” on page 6-9 for information on locating a rule.
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.



Rulesheet: MyBuyBeans

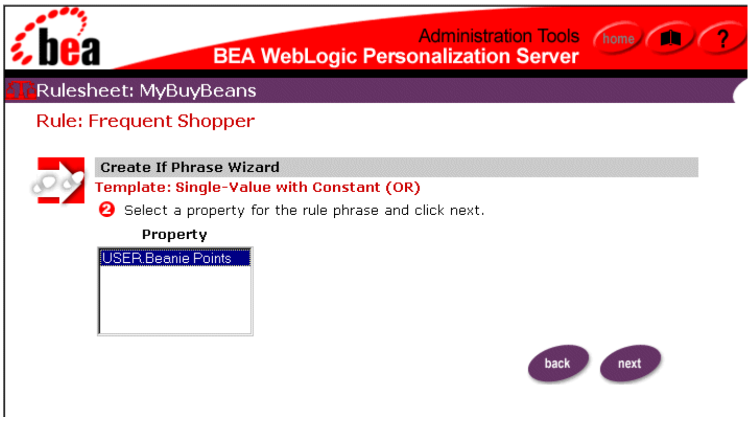
Rule: Frequent Shopper

Create If Phrase Wizard

1 Select a template for the rule phrase and click next.

[✓ Single-Value with Constant]	Example
This template compares a single-value property to a constant and allows you to add multiple OR conditions.	user.annual_income > 100,000 or 500,000

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



Rulesheet: MyBuyBeans

Rule: Frequent Shopper

Create If Phrase Wizard

Template: Single-Value with Constant (OR)

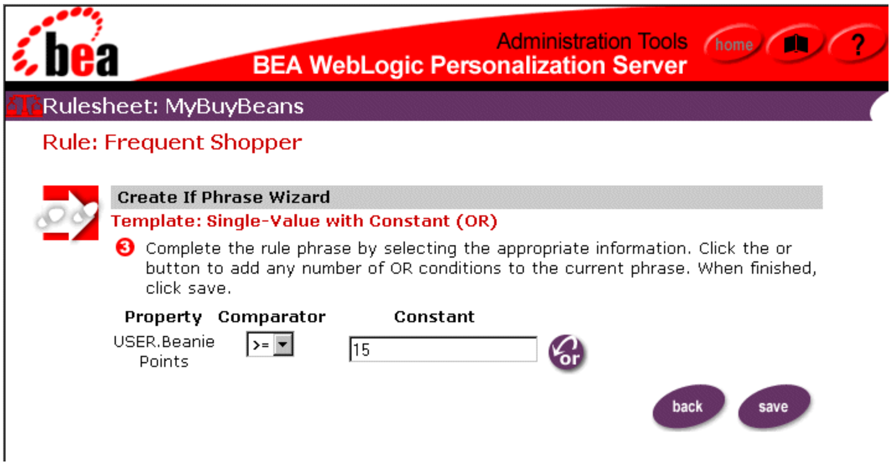
2 Select a property for the rule phrase and click next.

Property

USER.Beanie Points

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

7. Click on 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.



Rulesheet: MyBuyBeans

Rule: Frequent Shopper

Create If Phrase Wizard

Template: Single-Value with Constant (OR)

3 Complete the rule phrase by selecting the appropriate information. Click the or button to add any number of OR conditions to the current phrase. When finished, click save.

Property	Comparator	Constant
USER.Beanie Points	>=	15

8. Select a comparator from the Comparator drop-down list box and enter a value into the Constant field.

9. To add an or condition to the phrase, click the or 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 or 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.

Editing 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 rulesheet which contains the rule you want to edit. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule to edit. See “Finding a rule” on page 6-9 for information on locating a rule.
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

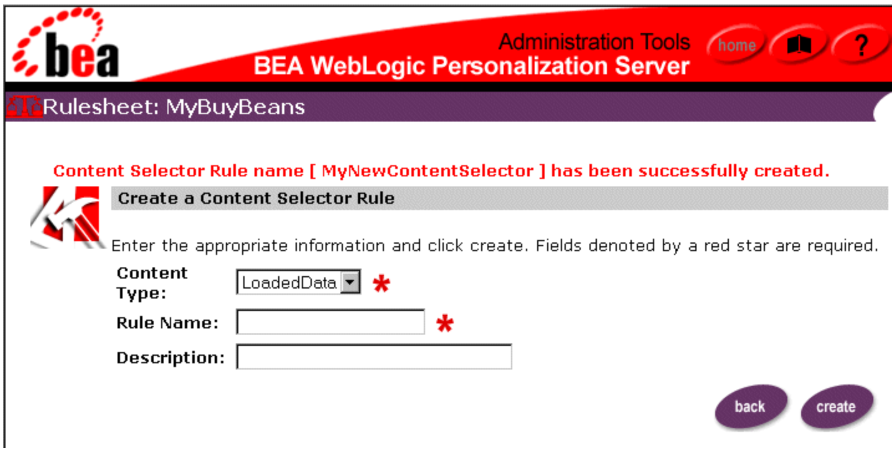
1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet which contains the rule to which you want to add a phrase. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule which contains the rule phrase to delete. See “Finding a rule” on page 6-9 for information on locating a rule.
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

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet in which you want to create the rule by clicking the rulesheet name. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet 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 rulesheet. 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 Rulesheet view page. The Rulesheet view page appears with the new rule(s) you created displayed in the alphabetical rule list.



Content Selector Rule name [MyNewContentSelector] has been successfully created.

Create a Content Selector Rule

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

Content Type: LoadedData *

Rule Name: *

Description:

Adding 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 rulesheet which contains the rule you want to edit. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule to edit. See "Finding a rule" on page 6-9 for information on locating a rule.
4. Click the name of the rule. The Rule view page appears.
5. Click **class** 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.

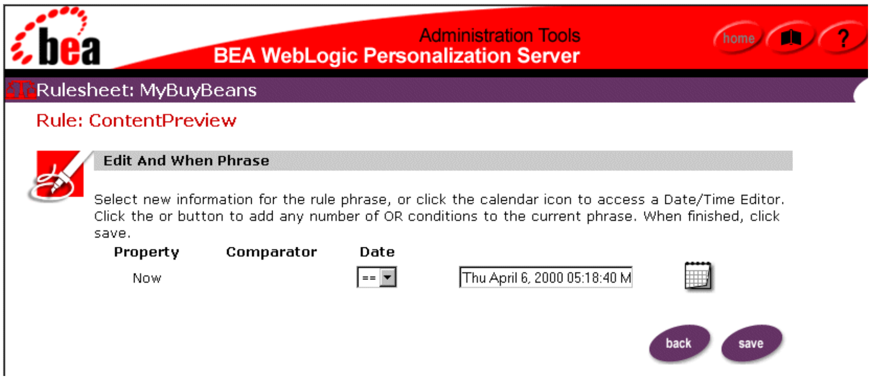
Note: The * character allows you to perform a wildcard search. Using the * character alone returns a list of all classifier rules.

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 in the beneath the If the user has the following characteristics bar.

Adding 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 rulesheet which contains the rule you want to edit. See “Creating a rulesheet” on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule to edit. See “Finding a rule” on page 6-9 for information on locating a rule.
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.



Rulesheet: MyBuyBeans

Rule: ContentPreview

Edit And When Phrase

Select new information for the rule phrase, or click the calendar icon to access a Date/Time Editor. Click the or button to add any number of OR conditions to the current phrase. When finished, click save.

Property	Comparator	Date
Now	==	Thu April 6, 2000 05:18:40 M

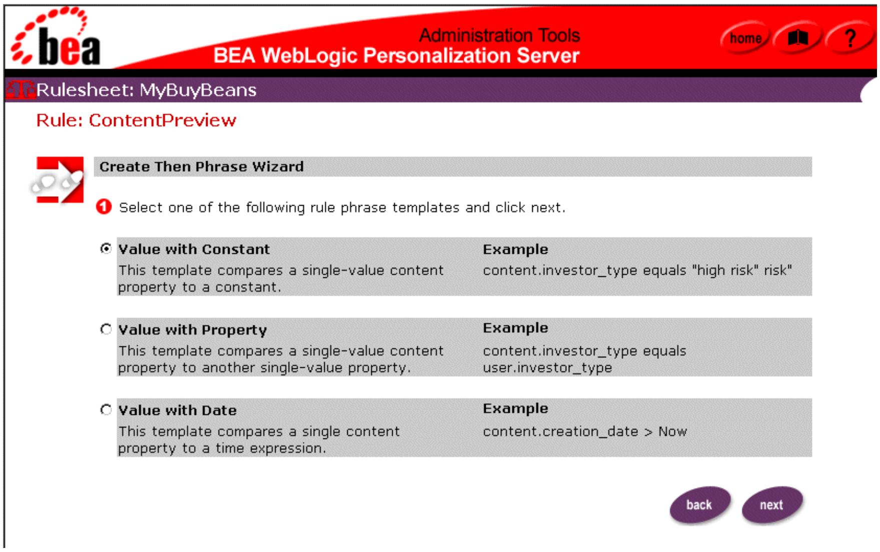
6. Select a comparator from the Comparator drop-down list box.
7. Select a date and time to compare using the icon to display a calendar or by entering a date into the Date field.

Note: You must include 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

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet which contains the rule you want to edit. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule to edit. See "Finding a rule" on page 6-9 for information on locating a rule.
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.



Rulesheet: MyBuyBeans

Rule: ContentPreview

Create Then Phrase Wizard

Select one of the following rule phrase templates and click next.

[✓ Value with Constant] This template compares a single-value content property to a constant.	Example content.investor_type equals "high risk" risk"
Value with Property This template compares a single-value content property to another single-value property.	Example content.investor_type equals user.investor_type
Value with Date This template compares a single content property to a time expression.	Example content.creation_date > Now

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

1. From the Administration Tools Home Page, click the Rules Management icon. The Rules Management home page appears.
2. Open the rulesheet which contains the rule you want to edit. See "Creating a rulesheet" on page 6-7 for instructions on creating a rulesheet. The Rulesheet view page appears.
3. Find the rule to edit. See "Finding a rule" on page 6-9 for information on locating a rule.
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 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.

Index