BEA WebLogic Portal™

User Management Guide

Version 8.1 with Service Pack 4
Document Revised: May 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

3. Unified User Profiles

4. Overview of Delegated Administration

5. Overview of Visitor Entitlements

About This Document

This document describes user management, delegated administration, and visitor entitlements, and shows you how to set up anonymous tracked users and unified user profiles.

This document covers the following topics:

- Chapter 1, "User Management Overview" describes user management in WebLogic Portal.
- Chapter 2, "Anonymous Users" describes the interactions that anonymous (non-authenticated) users can have in your portal applications and shows how to enable anonymous user tracking to provide personalization and track the behavior of non-authenticated users.
- Chapter 3, "Unified User Profiles" shows you how to create a unified user profile to access user properties (other than name and password) that are stored in external user stores.
- Chapter 4, "Overview of Delegated Administration" provides an overview of how you can use delegated administration to establish administrative privileges in your portal applications.
- Chapter 5, "Overview of Visitor Entitlements" provides an overview of how you can use visitor entitlements to control visitor access to your portals.

What You Need to Know

This document is intended for new or existing BEA partners who want to distribute products that include or interoperate with WebLogic Platform.

Product Documentation on the dev2dev Web Site

BEA product documentation, along with other information about BEA software, is available from the BEA dev2dev Web site:

`http://dev2dev.bea.com`

To view the documentation for a particular product, select that product from the list on the dev2dev page; the home page for the specified product is displayed. From the menu on the left side of the screen, select Documentation for the appropriate release. The home page for the complete documentation set for the product and release you have selected is displayed.

Related Information

Readers of this document may find the following documentation and resources especially useful:

- For general information about Java applications, go to the Sun Microsystems, Inc. Java Web site at `http://java.sun.com`.

Contact Us

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

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Portal ProductVersion.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support at **http://support.bea.com**. You can also contact Customer Support by using the contact information provided on the quick reference sheet titled “BEA Customer Support,” 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
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.
monospace text	Indicates *user input*, as shown in the following examples: • Filenames: `config.xml` • Pathnames: *`BEAHOME`*`/config/examples` • Commands: `java -Dbea.home=BEA_HOME` • Code: `public TextMsg createTextMsg(`
	Indicates *computer output*, such as error messages, as shown in the following example: `Exception occurred during event dispatching:java.lang.ArrayIndexOutOfBoundsException: No such child: 0`
monospace boldface text	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
monospace italic text	Identifies variables in code. *Example*: `String` *`expr`*
{ }	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*: `java utils.MulticastTest -n` *`name`* `[-p` *`portnumber`*`]`
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. *Example*: `java weblogic.deploy [list|deploy|update]`

Convention	Item
. . .	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 "file1.cpp file2.cpp file3.cpp . . ."`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER 1

User Management Overview

This guide describes the role of users in your portal application and provides instructions on adding and managing those users.

This chapter, which provides an overview of user management, includes the following sections:

Overview of Users and Groups

WebLogic Portal supports management of two primary types of users: portal administrators and portal end users (visitors). An *administrator* uses the WebLogic Administration Portal to manage portal content and portal users, and to build portals with existing portal resources. A *visitor* is an end user of the portals assembled by portal administrators.

As an administrator of users, you can combine users into user groups to simplify management and administration of users. Groups allow you to create logical groupings of users for ease of organization, more easily finding users, and for setting up delegated administration and visitor entitlements. The ability to group users allows you to deal with large sets of users at one time.

A child group is a subset of a higher-level group (parent), and you can model your organizational hierarchy into a collection of groups and subgroups to whatever depth you need. This makes it easy for you to structure your users into groups and subgroups that look like your organization, making it easier for you to manage those users. For more information about groups and subgroups, see the Overview of Group Hierarchy.

Administrators with full user/group management rights can add, remove, move, and change user profile properties on the users and groups in the WebLogic Administration Portal. User properties, such as address, phone number, social security number, and so on, can be used in setting up personalization and defining rules for Delegated Administration and Visitor Entitlements. For more on user information, see the Overview of User Profiles and Property Sets.

Multiple Authentication Providers

WebLogic Portal lets you access more than one user database (authentication provider), letting you select users and groups from multiple databases. For information on using more than one authentication provider, see "Using Multiple Authentication Providers in Portal Development" in the WebLogic Workshop help system at http://e-docs.bea.com/workshop/docs81/doc/en/portal/security/securityMap.html.

Overview of Group Hierarchy

A *group*, also known as a *user-group*, is a named collection of users in the user management system. Groups provide a convenient way to refer to a related group of users, such as a department, team, or regional office.

Because groups can belong to other groups, they provide a way to easily sub-classify users with a group hierarchy tailored to fit your needs.

For example, you might have a top-level group called AllEmployees that contains all of the employees in your company. The AllEmployees group might then contain other groups, each of which contains just the employees of offices in various geographic locations:

- AllEmployees (group containing all employees)
  - NewYork (group containing just employees in New York)
    - Executive
    - Marketing
    - Sales
    - Custodial

- SanFrancisco (group containing just employees in San Francisco)
- Seattle (group containing just employees in Seattle)
- Denver (group containing just employees in Denver)

Overview of User Profiles and Property Sets

A *user profile* is a schema that determines which data you collect and store about a user. Similarly, a *group profile* is a schema that determines which data you collect and store about a specific group of users. Each piece of data in a user profile is called a user property. A user profile is the entire collection of user property values for a user from all available user property sets.

User properties 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.

User profiles use *property sets* to organize the properties that they contain. A property set is a convenient way to give a name to a group of properties for a specific purpose.

A *property set type* establishes a set of expectations for how a property set is used. For example, a property set for users called “personal” could contain such properties as age, gender, marital status, and social security number. Another property set called “preferences” could contain such properties as hobby, favorite color, and news preference.

User profiles are a key component in Interaction Management. When users log in to a portal, the portal knows all their property values and can target them with personalized content, e-mails, and discounts based on the personalization rules you set up.

Security and User Profiles

User profiles are a key component in portal security. When you use visitor entitlements to limit end user access to portal resources (desktops, books, pages, portlets, and other resources), the visitor entitlements can be defined with user properties. For example, you could create an visitor entitlement role called “manager” that says, “If a user who logs in has an “employee_type” property with a value of “manager,” that user belongs to the “manager” role. You could then select a portlet and set its visitor entitlement to the “manager” role. Then, when a manager logs in, the manager sees the portlet. If a non-manager logs in, the person does not see the portlet.

User profiles also play a role in setting up delegated administration. You can create delegated administration roles using, among other things, user properties.

Note: When setting up visitor entitlements and delegated administration in the WebLogic Administration Portal, you will see the following option: "The user has specific characteristics." This is the option you use to select user profile properties in defining the visitor entitlement or delegated administration role.

For more information, see Visitor Entitlements and Delegated Administration in this guide.

Creating User Properties and Users

This section describes how to create the properties you want to make part of each user's profile, then describes different ways to add users to the system.

Creating User Profile Properties

To create properties that will be part of the profile for each user (and group), use the User Profile Property Designer in WebLogic Workshop. For more information, see "Creating User Profile Properties" in the WebLogic Workshop help system at http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/UserProfileProperties.html.

After you have deployed your portal application to production, any modifications you make to user profile properties in WebLogic Workshop, you must redeploy the EAR file to the production server. For detailed information on EAR deployment, see the Production Operations User Guide.

Creating Users

There are different ways of setting up or identifying users in your portals. You are likely to use many of these ways:

- Adding Users with the WebLogic Administration Portal
- Adding Users with the WebLogic Administration Console
- Letting Users Add Themselves

Adding Users with the WebLogic Administration Portal

You can add users and organize them into groups in your domain using the WebLogic Administration Portal.

To access the WebLogic Administration Portal:

On the production server: With the server running, enter the following URL in a browser: http://<server>:<port>/<portalApp>Admin. For example, http://localhost:7001/greatAppAdmin.

On your development server: You can use the same method as used on the production server, or you can use WebLogic Workshop. With your portal application open and the development server running, choose Portal --> Portal Administration.

Once in the WebLogic Administration Portal, select the "Users & Groups" tool to add users and groups.

For detailed instructions on adding users through the WebLogic Administration Portal, see the WebLogic Administration Portal help system's "Users and Groups" topics at http://edocs.bea.com/wlp/docs81/adminportal/index.html#usersgroups.

Adding Users with the WebLogic Administration Console

You can add users and organize them into groups in your domain using the WebLogic Administration Console. Add users this way for convenience if you work frequently in the WebLogic Administration Console rather than in the WebLogic Administration Portal.

For instructions on creating users and groups, see Security in the WebLogic Administration Console help system documentation.

To access the WebLogic Administration Console:

On the production server: With the server running, enter the following URL in a browser: http://<server>:<port>/console. For example, http://localhost:7001/console.

On your development server: You can use the same method as used on the production server, or you can use WebLogic Workshop. With the development server running, choose Tools --> WebLogic Server --> WebLogic Console.

Once in the WebLogic Administration Console, go to Security --> Realms --> [your_realm] and select Users or Groups to add either.

Letting Users Add Themselves

You can build functionality into your portal application that lets users add themselves to the domain. Use any of the following features:

<ugm:createUser> - The WebLogic Workshop Portal Extensions <ugm:createUser> JSP tag lets you add users to the domain. See the WebLogic Workshop help system at http://e-docs.bea.com/workshop/docs81/doc/en/portal/taglib/www.bea.com/servers/p13n/tags/userGroupManagement/createUser.html.

User Provider Control - The Create User control lets you add user-creation functionality to Java Page Flows and surface that functionality in portlets. See the WebLogic Workshop help system

at http://e-docs.bea.com/workshop/docs81/doc/en/portal/controls/portalcontrols/ctrlUserProvider.html.

WebLogic Portal API - The WebLogic Workshop Portal Extensions API provides user management classes for creating users. See the com.bea.p13n.controls.ejb.usermgmt.UserManager class in the WebLogic Portal Javadoc at http://edocs.bea.com/wlp/docs81/javadoc.

Managing Anonymous and External Users

For information on working with anonymous users, see Anonymous Users in this guide.

For information on accessing user profile properties stored outside of WebLogic Portal, see Unified User Profiles in this guide.

CHAPTER 2

Anonymous Users

When a user visits your portal without logging in, that user is called “anonymous.” While working with anonymous users is inherently limiting in some ways (because there are few things you know for sure about them).

But there are some things you can know for sure about users, such as the date and time they are visiting and any information in the request or session.

WebLogic Portal takes advantage of that information and provides many features that make working with anonymous users more useful and meaningful in your portals, such as letting you set up visitor entitlements, delegated administration, and some types of personalization based on that information.

WebLogic Portal provides even more power in working with anonymous users with a feature called anonymous user tracking. When you use anonymous user tracking, anonymous users are given records in a database, where you can store meaningful information about them such as personal preferences.

Note: Campaigns and behavior tracking are not currently supported for anonymous, non-trackable users.

This chapter includes the following sections:

Setting up Anonymous User Tracking

WebLogic Portal lets you identify and retain information about non-authenticated visitors to your portals. When you enable anonymous user tracking, non-authenticated users receive a cookie after a predetermined time (30 seconds by default), and preference information is persisted in a database rather than in memory.

Each time an anonymous tracked user returns to the portal, the ID in their cookie matches the primary key in the tracked anonymous user database, their previous user properties are maintained, and your personalization and campaign rules will work for those users. When anonymous tracked users register in your portal, their user profile is moved from the anonymous tracked user database to the user database.

If users do not have cookies enabled or if they delete cookies frequently, there is no way to match the users with their existing records in the anonymous tracked user database, and on subsequent returns to the portal these users are treated as new anonymous users.

Tracking Anonymous Users

This feature enables you to track users before they register with your site, meaning you can present personalized content to an anonymous user over multiple sessions. You can also persist information about a user's behavior, provided cookies are enabled on the user's browser. This feature can be parameterized to exclude those who spend less than a designated period of time on your site, avoiding the expense of storing data on irrelevant users. Finally, should a tracked anonymous user choose to register with your site, the data collected on that user is persisted in the account of the newly-registered user.

Five User Profile Types for User Tracking

If Anonymous User Tracking is not enabled, users who visit the site without registering will be considered ANONYMOUS. If it is enabled, such users would be considered TRACKABLE on the first visit.

If a TRACKABLE anonymous user remains at a site longer than the specified duration, the user is converted automatically into a TRACKED anonymous user, and a cookie with the tracking id is given to the user. That id is used as the primary key of the tracking EJBs associated with that user, so that when the session ends, the user’s properties are persisted and available for the next visit.

If this TRACKED anonymous user returns, the tracking id in the user's cookie is used to retrieve the user properties from the database, and those properties may be used to enable campaigns to be run against the user.

Finally, if a TRACKED anonymous user registers at the site, the tracking data is transferred to the newly created user (now considered a REGISTERED user) and deleted from the anonymous user.

The final user profile type is UNKNOWN, which designates users whose status cannot be determined.

Tracking Based on Visit Duration

Because users don't always remain at a site long enough to demonstrate interest, the duration of the initial visit is used as a configurable trigger. If this value is set to 30 seconds, a user that leaves after 25 seconds would not be remembered on the next visit to the same site. A 30-second or more user would be remembered, even without having registered.

This parameter is set by editing the web.xml file (Enabling Anonymous User Tracking).

Enabling Anonymous User Tracking

Here are specific instructions for enabling your Web application to track anonymous users. Using this feature requires that cookies be enabled on the user's browser.

Anonymous User Tracking is not enabled by default. It must be enabled in order to function by completing the following tasks:

Enable Anonymous user Duration Checking

Configure Visit Duration

To Enable Tracked Anonymous Users

Anonymous User Tracking is implemented by configuring the PortalServletFilter by adding the following entry to the web.xml file inside your Web application:

```
<!-- Portal Servlet Filter, always required for Portal -->

    <filter>

    <filter-name> PortalServletFilter </filter-name>

    <filter-class> com.bea.p13n.servlets.PortalServletFilter
</filter-class>
```

```
    <init-param>
        <param-name>fireSessionLoginEvent</param-name>
        <param-value>false</param-value>
       <description> Option to fire SessionLoginEvent , defaults to false
if not set</description>
    </init-param>
    <init-param>
        <param-name>createAnonymousProfile</param-name>
        <param-value>true</param-value>
        <description> Filter will create an anonymous profile for every
session. Defaults to true if not set</description>
    </init-param>
    <init-param>
        <param-name>enableTrackedAnonymous</param-name>
        <param-value>true</param-value>
        <description> Option to track anonymous users , defaults to false
if not set. 'createAnonymousProfile' is ignored if this is
true</description>
    </init-param>
    <init-param>
        <param-name>trackedAnonymousVisitDuration</param-name>
        <param-value>5</param-value>
        <description> Length in seconds visitor must be on site before we
start tracking them . Defaults to 60 seconds if not set</description>
</init-param>
</filter>
```

Creating Anonymous User Profiles

If a user visits your site and then registers within the same session, any data collected on that user is persisted in the account of the newly-registered user.

The Create Anonymous Profile parameter does not enable Anonymous Users to be tracked from one session to the next without registering. This would require the Tracked Anonymous User feature.

NOTE: This feature defaults to True.

Disabling the Create Anonymous Profile Feature

This feature is configured with the PortalServletFilter settings in the web.xml for the enterprise application.

To disable this feature, set the init-param node called createAnonymousProfile to False.

```
<init-param>
    <param-name>createAnonymousProfile</param-name>
        <param-value>true</param-value>
            <description>
            Filter will create an anonymous profile for every session.
            Defaults to true if not set.
            </description>
</init-param>
```

Handling Anonymous (Non-tracked) Users

You can target anonymous (non-tracked) users with personalized content using any personalization tool besides campaigns and behavior tracking--tools such as default (non-campaign) placeholders and content selectors. Campaigns and behavior tracking are not currently supported for anonymous, non-tracked users.

The main difference between completely anonymous (non-tracked) users and registered or anonymous tracked users is how long their profile information is retained.

For example, if an anonymous user visits a portal and sets her preferences to “favorite color=purple” and “favorite hobby=reading,” those values are persisted in memory and can be used to display personalized content and trigger campaigns while the browser session lasts. However, if the user closes the browser and revisits the portal, she has to re-enter her user preferences. Registered and anonymous tracked users have their preferences saved in a database.

While anonymous users do not retain a consistent store of user preferences from session to session, you can still provide a fair amount of interaction management functionality. For example, many personalization conditions are based on such things as generic HTTP session and request properties you define, dates and times, and personal session preferences that have no relationship to registered users and their profile properties. For example, you could display personalized content for anonymous users accessing your portal with a specific type of browser during a specific time frame.

CHAPTER 3

Unified User Profiles

If you have an existing store of users, groups, and additional properties (such as address, e-mail address, phone number, and so on), unified user profiles are a necessary part of bringing those user properties into the WebLogic Portal environment, where they can be used for retrieving and editing property values and setting up personalization, delegated administration, and visitor entitlements.

This chapter describes the unified user profile, when to use it, and when not to use it.

Note: This topic contains the terms “user store” and “data store.” A user store can contain users and groups, as well as additional properties. A data store implies that the store does not have to contain users and groups. It can simply contain properties.

This chapter includes the following sections:

Overview of Unified User Profiles

Here is an example that explains what a unified user profile is and does:

Let's say you're creating a new portal application that you want users to be able to log in to. Let's also say your users are stored in an RDBMS user store outside of the WebLogic environment. You could connect WebLogic Server (your portal application's domain server instance) to your

RDBMS system, and your users could log in to your portal application as if their user names and passwords were stored in WebLogic Server. If authentication was all you wanted to provide through your RDBMS user store, you could stop here without needing a unified user profile.

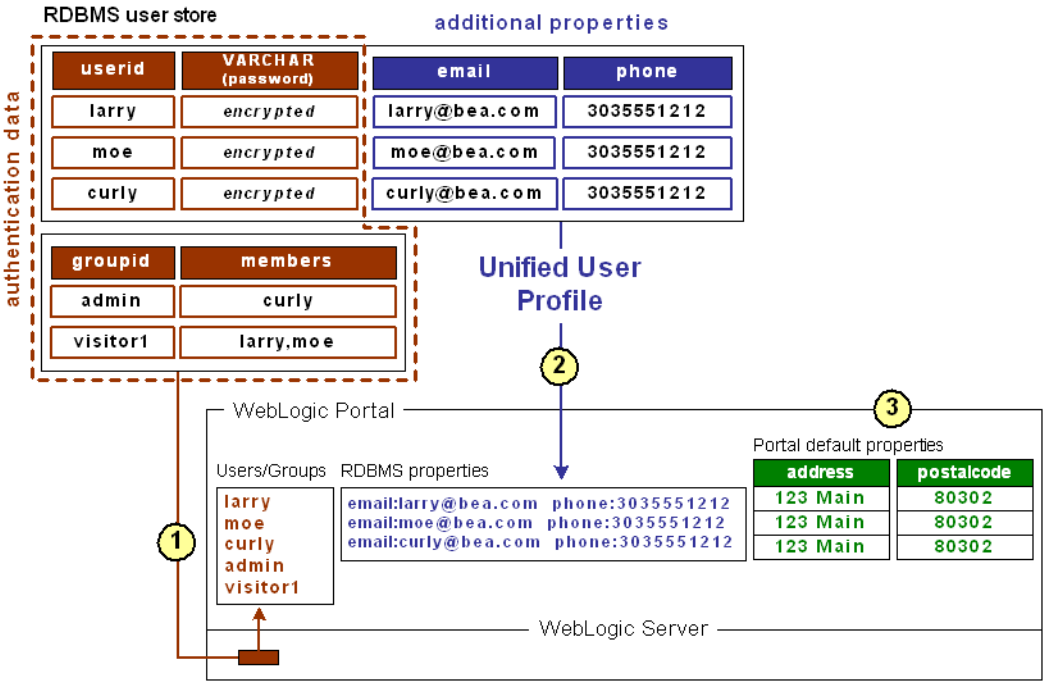
However, let's say you also stored e-mail and phone number information (properties) for users in your RDBMS user store, and you wanted to be able to access those properties in your portal applications. In this case, you need to create a unified user profile for your RDBMS user store that lets you access those additional properties from your code.

Technically speaking, a unified user profile is a stateless session bean you create (with associated classes) that lets WebLogic Portal read property values stored in external data stores, such as LDAP servers and databases. Once connected to an external data store with a unified user profile, you can use portal JSP tags, controls, and the WebLogic Portal API to retrieve user property values from that store. You can also take the extra step of surfacing these external properties in the WebLogic Administration portal, where the properties can be used to define rules for personalization, visitor entitlements, and delegated administration.

Whether or not you have additional properties stored in your external user store, the external users and groups you connect to WebLogic Server are automatically assigned the default user property values you have set up in WebLogic Portal, without the use of a unified user profile. With the WebLogic Administration Portal, you can change the default WebLogic Portal property values for those users. These values are stored in WebLogic Portal's RDBMS data store using the Portal schema.

The following figure shows where a unified user profile fits between an external user store and the WebLogic environment.

Figure 3-1 Unified User Profile



1	This external RDBMS user store, which supports authentication, contains users (principals) and passwords in one database table and groups (principals) in another. Giving a user store authentication capabilities (as an authentication provider or identity asserter) involves configuration steps not associated with the unified user profile configuration process. (See Developing Security Providers for WebLogic Server.) Unified user profile configuration is not dependent on the authentication provider configuration and vice versa. Once the RDBMS authentication provider is connected to WebLogic Server, WebLogic Server (and WebLogic Portal) can see those users and groups. Those users can log in to your portal applications, and you can include those users and groups in your rules for personalization, delegated administration, and visitor entitlements. Also, WebLogic Portal's ProfileWrapper maps the principals to properties kept in the Portal schema, thereby establishing the user profile.
2	Unified User Profile - The same external table that contains users and passwords also contains additional properties (email and phone) for each user. These additional properties are not part of authentication; but they are part of each user's profile. If you want to access these properties in your portal applications (with the WebLogic Portal JSP tags, controls, or API), you must create a unified user profile for the RDBMS user store. When you create the unified user profile, the ProfileWrapper includes the external properties in the user profile. The unified user profile consists of a stateless session bean and associated classes that you create. If you want to surface any of these properties in the WebLogic Administration Portal to be used in defining rules for personalization, delegated administration, or visitor entitlements, create a user profile property set for the external user store in addition to implementing your unified user profile session bean. The property set provides metadata about your external properties so that WebLogic Workshop and the WebLogic Administration Portal know how to display them. Properties from an external data store are typically read only in the WebLogic Administration Portal.
3	WebLogic Portal lets you create user/group properties and set default values for those properties. Any user or group in WebLogic Server, whether created in the default LDAP store or brought in through a connection to an external user store, is automatically assigned those default property values; and you can change the default values for each user or group, programmatically or in the WebLogic Administration Portal. This does not involve unified user profiles, because the properties to be retrieved are local, not stored in an external data store. In the illustration, after the authentication provider or identity asserter provides the principals, the ProfileWrapper combines the principals with the external properties of email and phone (retrieved by the unified user profile) and the default WebLogic Portal properties of address and postal code, all of which make up the full user profile.

What a Unified User Profile is Not

A user profile is not a security realm, and it does not provide authentication. It is not even the external user store itself. It is the connection (stateless session bean with associated classes) that lets you read properties in the external user store.

When Should You Create a Unified User Profile?

Create a unified user profile for an external data store if you want to do any of the following:

- Use WebLogic Portal's JSP tags, controls, or API to retrieve property values from that external store.
- Surface external properties in the WebLogic Administration Portal for use in defining rules for personalization, delegated administration, or visitor entitlements. Users and groups are not considered properties.

When Don't You Need a Unified User Profile?

You do not need to create a unified user profile for an external data store if you only want to:

- Provide authentication for users in the external user store.
- Define rules for personalization, delegated administration, or visitor entitlements based only on users or groups in an external user store, not on user properties.
- Define rules for personalization, delegated administration, or visitor entitlements based on the WebLogic Portal user profile properties you create in WebLogic Workshop, which are kept in the Portal schema.

Setting up Unified User Profiles

This topic provides guidelines and instructions on creating a unified user profile to access user/group properties from an external user store. (See Unified User Profiles Overview for overview information.)

To create a UUP to retrieve user data from external sources, complete the following tasks:

- Create an EntityPropertyManager EJB to Represent External Data
- Deploy a ProfileManager That Can Use the New EntityPropertyManager

- If you have an LDAP server for which you want to create a unified user profile, WebLogic Portal provides a default unified user profile you can modify. See Retrieving User Profile Data from LDAP.

Create an EntityPropertyManager EJB to Represent External Data

To incorporate data from an external source, you must first create a stateless session bean that implements the methods of the com.bea.p13n.property.EntityPropertyManager remote interface. EntityPropertyManager is the remote interface for a session bean that handles the persistence of property data and the creation and deletion of profile records. By default, EntityPropertyManager provides read-only access to external properties.

In addition, the stateless session bean should include a home interface and an implementation class. For example:

```
MyEntityPropertyManager

extends com.bea.p13n.property.EntityPropertyManager

MyEntityPropertyManagerHome

extends javax.ejb.EJBHome
```

Your implementation class can extend the EntityPropertyManagerImpl class. However the only requirement is that your implementation class is a valid implementation of the MyEntityPropertyManager remote interface. For example:

```
MyEntityPropertyManagerImpl extends

com.bea.p13n.property.internal.EntityPropertyManagerImpl
```

or

```
MyEntityPropertyManagerImpl extends

javax.ejb.SessionBean
```

Recommended EJB Guidelines

We recommend the following guidelines for your new EJB:

- Your custom EntityPropertyManager is not a default EntityPropertyManager. A default EntityPropertyManager is used to get/set/remove properties in the Portal schema. Your custom EntityPropertyManager does not have to support the following methods. It can throw java.lang.UsupportedOperationException instead:

  - getDynamicProperties()
  - getEntityNames()
  - getHomeName()
  - getPropertyLocator()
  - getUniqueId()

- If you want to be able to use the portal framework and tools to create and remove users in your external data store, you must support the createUniqueId() and removeEntity() methods. However, your custom EntityPropertyManager is not the default EntityPropertyManager so your createUniqueId() method does not have to return a unique number. It must create the user entity in your external data store and then it can return any number, such as -1.
- The following recommendations apply to the EntityPropertyManager() methods that you must support:
  - getProperty() - Use caching. You should support the getProperties() method to retrieve all properties for a user at once, caching them at the same time. Your getProperty() method should use getProperties().
  - setProperty() - Use caching.
  - removeProperties(), removeProperty() - After these methods are called, a call to getProperty() should return null for the property. Remove properties from the cache, too.
- Your implementations of the getProperty(), setProperty(), removeProperty(), and removeProperties() methods must include any logic necessary to connect to the external system.
- If you want to cache property data, the methods must be able to cache profile data appropriately for that system. (See the com.bea.p13n.cache package in the WebLogic Portal Javadoc.)
- If the external system contains read-only data, any methods that modify profile data must throw a java.lang.UnsupportedOperationException. Additionally, if the external data source contains users that are created and deleted by something other than your WebLogic Portal createUniqueId() and removeEntity() methods can simply throw an UnsupportedOperationException.
- To avoid class loader dependency issues, make sure that your EJB resides in its own package.

- For ease of maintenance, place the compiled classes of your custom EntityPropertyManager bean in your own JAR file (instead of modifying an existing WebLogic Portal JAR file).

Before you deploy your JAR file, follow the steps in the next section.

Deploy a ProfileManager That Can Use the New EntityPropertyManager

A “user type” is a mapping of a ProfileType name to a particular ProfileManager. This mapping is done in the UserManager EJB deployment descriptor.

To access the data in your new EntityPropertyManager EJB, you must do one of the following:

- Modifying the Existing ProfileManager Deployment Configuration - In most cases you will be able to use the default deployment of ProfileManager, the UserProfileManager. You will modify the UserProfileManager's deployment descriptor to map a property set and/or properties to your custom EntityPropertyManager. If you support the createUniqueId() and removeEntity() methods in your custom EntityPropertyManager, you can use WebLogic Administration Portal to create a user of type “User” with a profile that can get/set properties using your custom EntityPropertyManager.
- Configuring and Deploying a New ProfileManager - In some cases you may want to deploy a newly configured ProfileManager that will be used instead of the UserProfileManager. This new ProfileManager is mapped to a ProfileType in the deployment descriptor for the UserManager. If you support the createUniqueId() and removeEntity() methods in your custom EntityPropertyManager, you can use the WebLogic Administration Portal (or API) to create a user of type “MyUser” (or anything else you name it) that can get/set properties using the customized deployment of the ProfileManager that is, in turn, configured to use your custom EntityPropertyManager.

ProfileManager is a stateless session bean that manages access to the profile values that the EntityPropertyManager EJB retrieves. It relies on a set of mapping statements in its deployment descriptor to find data. For example, the ProfileManager receives a request for the value of the “DateOfBirth” property, which is located in the “PersonalData” property set. ProfileManager uses the mapping statements in its deployment descriptor to determine which EntityPropertyManager EJB contains the data.

Modifying the Existing ProfileManager Deployment Configuration

If you use the existing UserProfileManager deployment to manage your user profiles, perform the following steps to modify the deployment configuration.

Under most circumstances, this is the method you should use to deploy your UUP. An example of this method is the deployment of the custom EntityPropertyManager for LDAP property retrieval, the LdapPropertyManager. The classes for the LdapPropertyManager are packaged in p13n_ejb.jar. The deployment descriptor for the UserProfileManager EJB is configured to map the "ldap" property set to the LdapPropertyManager. The UserProfileManager is deployed in p13n_ejb.jar.

1. Back up the p13n_ejb.jar file in your enterprise application root directory.
2. From p13n_ejb.jar, extract META-INF/ejb-jar.xml and open it for editing.
3. In ejb-jar.xml, find the <env-entry> element, as shown in the following example:

   ```
   <!-- map all properties in property set ldap to ldap server -->
   <env-entry>
       <env-entry-name>PropertyMapping/ldap</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>LdapPropertyManager</env-entry-value>
   </env-entry>
   ```

   and add an <env-entry> element after this to map a property set to your custom EntityPropertyManager, a shown in the following example:

   ```
   <!-- map all properties in UUPExample property set to
   MyEntityPropertyManager -->
   <env-entry>
       <env-entry-name>PropertyMapping/UUPExample</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>MyEntityPropertyManager</env-entry-value>
   </env-entry>
   ```

4. In ejb-jar.xml, find the <ejb-ref> element shown in the following example:

   ```
   <!-- an ldap property manager -->
   <ejb-ref>
       <ejb-ref-name>ejb/LdapPropertyManager</ejb-ref-name>
       <ejb-ref-type>Session</ejb-ref-type>
       <home>com.bea.p13n.property.EntityPropertyManagerHome</home>
       <remote>com.bea.p13n.property.EntityPropertyManager</remote>
   ```

```
</ejb-ref>
```

and add an <ejb-ref> element after this to map a reference to an EJB that matches the name from the previous step with ejb/ prepended as shown in the following example:

```
<!-- an example property manager -->
<ejb-ref>
    <ejb-ref-name>ejb/MyEntityPropertyManager</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>examples.usermgmt.MyEntityPropertyManagerHome</home>
    <remote>examples.usermgmt.MyEntityPropertyManager</remote>
</ejb-ref>
```

The home and remote class names match the classes from your EJB JAR file for your custom EntityPropertyManager.

5. If your EntityPropertyManager implementation handles creating and removing profile records, you must also add Creator and Remover entries. For example:

```
<env-entry>
    <env-entry-name>Creator/Creator1</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>MyEntityPropertyManager</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>Remover/Remover1</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>MyEntityPropertyManager</env-entry-value>
</env-entry>
```

This instructs the UserProfileManager to call your custom EntityPropertyManager when creating or deleting user profile records. The names “Creator1” and “Remover1” are arbitrary. All Creators and Removers will be iterated through when the UserProfileManager creates or removes a user profile. The value for the Creator and Remover matches the ejb-ref-name for your custom EntityPropertyManager without the ejb/ prefix.

6. From p13n_ejb.jar, extract META-INF/weblogic-ejb-jar.xml and open it for editing.
7. In weblogic-ejb-jar.xml, find the elements shown in the following example:

```
<weblogic-enterprise-bean>
    <ejb-name>UserProfileManager</ejb-name>
    <reference-descriptor>
        <ejb-reference-description>
            <ejb-ref-name>ejb/EntityPropertyManager</ejb-ref-name>
            <jndi-name>${APPNAME}.BEA_personalization.
EntityPropertyManager</jndi-name>
        </ejb-reference-description>
```

and add an ejb-reference-description to map the ejb-ref for your custom EntityPropertyManager to the JNDI name. This JNDI name must match the name you assigned in weblogic-ejb-jar.xml in the JAR file for your customer EntityPropertyManager. It should look like the following example:

```
<weblogic-enterprise-bean>
    <ejb-name>UserProfileManager</ejb-name>
    <reference-descriptor>
        <ejb-reference-description>
            <ejb-ref-name>ejb/EntityPropertyManager</ejb-ref-name>
            <jndi-name>${APPNAME}.BEA_personalization.
EntityPropertyManager</jndi-name>
        </ejb-reference-description>
        <ejb-reference-description>
            <ejb-ref-name>ejb/MyEntityPropertyManager</ejb-ref-name>
            <jndi-name>${APPNAME}.BEA_personalization.
MyEntityPropertyManager</jndi-name>
        </ejb-reference-description>
```

Note the ${APPNAME} string substitution variable. The WebLogic EJB container automatically substitutes the enterprise application name to scope the JNDI name to the application.

8. Update p13n_ejb.jar for your new deployment descriptors. You can use the jar uf command to update the modified META-INF/ deployment descriptors.

9. Edit your application's META-INF/application.xml to add an entry for your custom EntityPropertyManager EJB module as shown in the following example:

```
<module>
```

```
        <ejb>UUPExample.jar</ejb>
    </module>
```

10. If you are using an application-wide cache, you can manage it from the WebLogic Administration Console if you add a <Cache> tag for your cache to the META-INF/application-config.xml deployment descriptor for your enterprise application like this:

    ```
    <Cache Name="UUPExampleCache" TimeToLive="60000"/>
    ```

11. Verify the modified p13n_ejb.jar and your custom EntityPropertyManager EJB JAR archive are in the root directory of your enterprise application and start WebLogic Server.

12. Use the WebLogic Server Administration Console to verify your EJB module is deployed to the enterprise application and then use the console to add your server as a target for the EJB module. You need to select a target to have your domain's config.xml file updated to deploy your EJB module to the server.

13. Use the WebLogic Workshop Property Set Designer to create a User Profile (property set) that matches the name of the property set that you mapped to your custom EntityPropertyManager in ejb-jar.xml for the UserProfileManager (in p13n_ejb.jar). You could also map specific property names in a property set to your custom EntityPropertyManager, which would allow you to surface the properties and their values in the WebLogic Administration Portal for use in creating rules for personalization, delegated administration, and visitor entitlements.

Your new Unified User Profile type is ready to use. You can use the WebLogic Administration Portal to create a user, and it will use your UUP implementation when the “UUPExample” property set is being modified. When you call createUser(“bob”, “password”) or createUser(“bob”, “password”, null) on the UserManager, several things will happen:

- A user named “bob” is created in the security realm.
- A WebLogic Portal Server profile record is created for “bob” in the user store.
- If you set up the Creator mapping, the UserManager will call the default ProfileManager deployment (UserProfileManager) which will call your custom EntityPropertyManager to create a record for Bob in your data source.
- Retrieving Bob's profile will use the default ProfileManager deployment (UserProfileManager), and when you request a property belonging to the “UUPExample” property set, the request will be routed to your custom EntityPropertyManager implementation.

Configuring and Deploying a New ProfileManager

If you are going to deploy a newly configured ProfileManager instead of using the default ProfileManager (UserProfileManager) to manage your user profiles, perform the following steps to modify the deployment configuration. In most cases, you will not have to use this method of deployment. Use this method only if you need to support multiple types of users that require different ProfileManager deployments—deployments that allow a property set to be mapped to different custom EntityPropertyManagers based on ProfileType.

An example of this method is the deployment of the custom CustomerProfileManager in customer.jar. The CustomerProfileManager is configured to use the custom EntityPropertyManager (CustomerPropertyManager) for properties in the "CustomerProperties" property set. The UserManager EJB in p13n_ejb.jar is configured to map the "WLCS_Customer" ProfileType to the custom deployment of the ProfileManager, CustomerProfileManager.

To configure and deploy a new ProfileManager, use this procedure.

1. Back up the p13n_ejb.jar file in your enterprise application root directory.
2. From p13n_ejb.jar, extract META-INF/ejb-jar.xml, and open it for editing.
3. In ejb-jar.xml, copy the entire <session> tag for the UserProfileManager, and configure it to use your custom implementation class for your new deployment of ProfileManager.

   In addition, you could extend the UserProfileManager home and remote interfaces with your own interfaces if you want to repackage them to correspond to your packaging (for example., examples.usermgmt.MyProfileManagerHome, examples.usermgmt.MyProfileManager).

   However, it is sufficient to replace the bean implementation class:

   You must create an <env-entry> element to map a property set to your custom EntityPropertyManager. You must also create a <ejb-ref> element to map a reference to an EJB that matches the name from the PropertyMapping with ejb/ prepended. The home and remote class names for your custom EntityPropertyManager match the classes from your EJB JAR file for your custom EntityPropertyManager.

   Also, if your EntityPropertyManager implementation handles creating and removing profile records, you must also add Creator and Remover entries. This instructs your new ProfileManager to call your custom EntityPropertyManager when creating or deleting user profile records.

   Note: The name suffixes for the Creator and Remover, "Creator1" and "Remover1", are arbitrary. All Creators and Removers will be iterated through when your ProfileManager creates or removes a user profile. The value for the Creator and

Remover matches the <ejb-ref-name> for your custom EntityPropertyManager without the ejb/ prefix.

4. In ejb-jar.xml, you must add an <ejb-ref> to the UserManager EJB section to map your ProfileType to your new deployment of the ProfileManager, as shown in the following example:

```
<ejb-ref>
    <ejb-ref-name>ejb/ProfileType/UUPExampleUser</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.bea.p13n.usermgmt.profile.ProfileManagerHome</home>
    <remote>com.bea.p13n.usermgmt.profile.ProfileManager</remote>
</ejb-ref>
```

   The <ejb-ref-name> must start with ejb/ProfileType/ and must end with the name that you want to use as the profile type as an argument in the createUser() method of UserManager.

5. From p13n_ejb.jar, extract META-INF/weblogic-ejb-jar.xml and open it for editing.

6. In weblogic-ejb-jar.xml, copy the <weblogic-enterprise-bean> tag, shown in the following example, for the UserProfileManager and configure it for your new ProfileManager deployment:

```
<weblogic-enterprise-bean>
    <ejb-name>MyProfileManager</ejb-name>
    <reference-descriptor>
        <ejb-reference-description>
            <ejb-ref-name>ejb/EntityPropertyManager</ejb-ref-name>
            <jndi-name>${APPNAME}.BEA_personalization.
EntityPropertyManager</jndi-name>
        </ejb-reference-description>
        <ejb-reference-description>
            <ejb-ref-name>ejb/PropertySetManager</ejb-ref-name>
            <jndi-name>${APPNAME}.BEA_personalization.
PropertySetManager</jndi-name>
        </ejb-reference-description>
        <ejb-reference-description>
            <ejb-ref-name>ejb/MyEntityPropertyManager</ejb-ref-name>
```

```
            <jndi-name>${APPNAME}.BEA_personalization.
MyEnitityPropertyManager</jndi-name>

        </ejb-reference-description>

    </reference-descriptor>

    <jndi-name>${APPNAME}.BEA_personalization.
MyProfileManager</jndi-name>

</weblogic-enterprise-bean>
```

You must create an <ejb-reference-description> to map the <ejb-ref> for your custom EntityPropertyManager to the JNDI name. This JNDI name must match the name you assigned in weblogic-ejb-jar.xml in the JAR file for your custom EntityPropertyManager.

Note the ${APPNAME} string substitution variable. The WebLogic Server EJB container automatically substitutes the enterprise application name to scope the JNDI name to the application.

7. In weblogic-ejb-jar.xml, copy the <transaction-isolation> tag for the UserProfileManager, shown in the following example, and configure it for your new ProfileManager deployment:

```
<transaction-isolation>

    <isolation-level>TRANSACTION_READ_COMMITTED</isolation-level>

    <method>

        <ejb-name>MyProfileManager</ejb-name>

        <method-name>*</method-name>

    </method>

</transaction-isolation>
```

8. Create a temporary p13n_ejb.jar for your new deployment descriptors and your new ProfileManager bean implementation class. This temporary EJB JAR archive should not have any container classes in it. Run ejbc to generate new container classes.

9. Edit your application's META-INF/application.xml to add an entry for your custom EntityPropertyManager EJB module, as shown in the following example:

```
<module>

    <ejb>UUPExample.jar</ejb>

</module>
```

10. If you are using an application-wide cache, you can manage it from the WebLogic Server Administration Console if you add a <Cache> tag for your cache to the META-INF/application-config.xml deployment descriptor for your enterprise application as shown in the following example:

    ```
    <Cache Name="UUPExampleCache" TimeToLive="60000"/>
    ```

    Verify the modified p13n_ejb.jar and your custom EntityPropertyManager EJB JAR archive are in the root directory of your enterprise application and start your server.

11. Use the WebLogic Server Administration Console to verify your EJB module is deployed to the enterprise application and add your server as a target for the EJB module. You must select a target to have your domain's config.xml file updated to deploy your EJB module to the server.

12. Use the WebLogic Workshop Property Set Designer to create a User Profile (property set) that matches the name of the property set that you mapped to your custom EntityPropertyManager in ejb-jar.xml for the UserProfileManager (in p13n_ejb.jar). You could also map specific property names in a property set to your custom EntityPropertyManager, which would allow you to surface the properties and their values in the WebLogic Administration Portal for use in creating rules for personalization, delegated administration, and visitor entitlements.

Your new Unified User Profile type is ready to use. You can use the WebLogic Administration Portal to create a user, and it will use your UUP implementation when the “UUPExample” property set is being modified. That is because you mapped the ProfileType using an <ejb-ref> in your UserManager deployment descriptor, ejb/ProfileType/UUPExampleUser.

Now, when you call createUser(“bob”, “password”, “UUPExampleUser”) on the UserManager, several things will happen:

- A user named “bob” is created in the security realm.
- A WebLogic Portal Server profile record is created for “bob” in the WebLogic Portal RDBMS repository.
- If you set up the Creator mapping, the UserManager will call your new ProfileManager deployment, which will call your custom EntityPropertyManager to create a record for Bob in your data source.
- Retrieving Bob's profile will use your new ProfileManager deployment, and when you request a property belonging to the “UUPExample” property set, the request will be routed to your custom EntityPropertyManager implementation.

Retrieving User Profile Data from LDAP

WebLogic Portal provides a default unified user profile for retrieving properties from an LDAP server. Use this procedure to implement the LDAP unified user profile for retrieving properties from your LDAP server.

The LdapRealm security realm and the LdapPropertyManager unified user profile (UUP) for retrieving user properties from LDAP are independent of each other. They do not share configuration information and there is no requirement to use either one in conjunction with the other. A security realm has nothing to do with a user profile. A security realm provides user/password data, user/group associations, and group/group associations. A user profile provides user and group properties. A password is not a property.

In order to successfully retrieve the user profile from the LDAP server, ensure that you've done the following:

1. If you have already deployed the application on a WebLogic Portal instance, stop the server.
2. Extract p13n_ejb.jar from your application root to a temporary directory.
3. In the temporary directory, open META-INF/ejb-jar.xml, which contains a commented block called “Ldap Property Manager.” Uncomment and reconfigure this section using the following steps:

   a. Remove the closing comment mark (-->) from the end of the “Ldap Property Manager” block, just before the “Property Set Web Service EJB” block, and add it to the end of the first paragraph of the Ldap Property Manager block, like this:

   ```
   <!-- Ldap Property Manager
   To use this, uncomment it here as well as in weblogic-ejb-jar.xml.
   Configure the LDAP connection and settings using the env-entry values
   (see descriptions below). Do not forget to uncomment the ejb-link and
   method-permission tags for the LdapPropertyManager. An easy way to
   ensure you don't miss anything is to search for "ldap"
   (case-insensitive) here AND in weblogic-ejb-jar.xml. Search from the
   beginning to the end of the file.
   -->
   ```

   b. In the “Ldap Property Manager” block, look for the following default settings and replace them with your own:

ldap://server.company.com:389	Change this to the value of your LDAP server URL.
uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot	Change this to the value of your LDAP server's principal.
<env-entry-value>weblogic</env-entry-value>	Change “weblogic” to your LDAP server's principalCredential.
ou=People,o=company.com	Change this to your LDAP server's UserDN.
ou=Groups,o=company.com	Change this to your LDAP server's GroupDN.
<env-entry-value>uid</env-entry-value>	Change “uid” to your LDAP server's usernameAttribute setting.
<env-entry-value>cn</env-entry-value>	Change “cn” to your LDAP server's groupnameAttribute setting.

c. In the “User Profile Manager” and “Group Profile Manager” sections, find the following lines:

```
<!-- <ejb-link>LdapPropertyManager</ejb-link> -->
<ejb-link>EntityPropertyManager</ejb-link>
```

Uncomment the LdapPropertyManager line and delete the EntityPropertyManager line in both sections.

d. In the <method-permission> and <container-transaction> sections, find and uncomment the following:

```
<!--
<method>
    <ejb-name>LdapPropertyManager</ejb-name>
    <method-name>*</method-name>
</method>
-->
```

e. Check to see that you have uncommented all Ldap configurations by doing a search for “Ldap” in the file.

f. Save and close the file.

4. In the temporary directory, open META-INF/weblogic-ejb-jar.xml and perform the following modifications:

   a. Uncomment the “LdapPropertyManager” block: LdapPropertyManager

   ```
   <weblogic-enterprise-bean>
       <ejb-name>LdapPropertyManager</ejb-name>
       <enable-call-by-reference>True</enable-call-by-reference>

   <jndi-name>${APPNAME}.BEA_personalization.LdapPropertyManager</jndi-
   name>
   </weblogic-enterprise-bean>
   ```

   b. In the “Security configuration” section of the file, uncomment the LdapPropertyManager method:

   ```
   <method>
       <ejb-name>LdapPropertyManager</ejb-name>
       <method-name>*</method-name>
   </method>
   ```

   c. Check to see that you have uncommented all Ldap configurations by doing a search for “Ldap” in the file.

   d. Save and close the file.

5. Replace the original p13n_ejb.jar with the modified version.

   a. Rename the original p13n_ejb.jar to use it as a backup. For example, rename it to p13n_ejb.jar.backup.

   b. JAR the temporary version of p13n_ejb.jar to which you made changes. Name it p13n_ejb.jar.

   c. Copy the new JAR to your application's root directory.

6. Start the server and re-deploy the application.

7. The properties from your LDAP server are now accessible through the WebLogic Portal API, JSP tags, and controls.

If you want to surface the properties from your LDAP server in the WebLogic Administration Portal (for use in defining rules for personalization, delegated administration, and visitor

entitlements), create a user profile property set called ldap.usr, and create properties in the property set that exactly match the names of the LDAP properties you want to surface.

Enabling SUBTREE_SCOPE Searches for Users and Groups

The LdapPropertyManager EJB in p13n_ejb.jar allows for the inspection of the LDAP schema to determine multi-valued versus single-value LDAP attributes, to allow for multiple userDN/groupDN, and to allow for SUBTREE_SCOPE searches for users and groups in the LDAP server. Following are more detailed explanations:

The determination of multi-value versus single-value LDAP attributes allows a developer to configure the ejb-jar.xml deployment descriptor for the LdapPropertyManager EJB to specify that the LDAP schema be used to determine if a property is single- or multi-value.

To enable SUBTREE-SCOPE for users and groups:

1. Stop the server.
2. Extract p13n_ejb.jar from your application root directory to a temporary directory and edit the temporary META-INF/ejb-jar.xml by setting the following env-entries.

```
<!-- Flag to specify if LDAP attributes will be determined to be single value

or multi-value via the schema obtained from the attribute. If false,

then the attribute is stored as multi-valued (a Collection) only if it has

more than one value. Leave false unless you intend to use multi-valued LDAP

attributes that may have only one value. Using true adds overhead to check

the LDAP schema. Also, if you use true beware that most LDAP attributes are

multi-value. For example, iPlanet Directory Server 5.x uses multi-value for

givenName, which you may not expect unless you are familiar with LDAP schemas.

This flag will apply to property searches for all userDNs and all groupDNs. -->

<env-entry>

    <env-entry-name>config/detectSingleValueFromSchema</env-entry-name>

    <env-entry-type>java.lang.Boolean</env-entry-type>

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

```
</env-entry>
<!-- Value of the name of the attribute in the LDAP schema that is used
to determine single value or multi-value (RFC2252 uses SINGLE-VALUE).
This attribute in the schema should be true for single value and false
or absent from the schema otherwise. The value only matters if
config/detectSingleValueFromSchema is true. -->
<env-entry>
    <env-entry-name>config/singleValueSchemaAttribute</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>SINGLE-VALUE</env-entry-value>
</env-entry>
```

It is not recommended that true be used for config/detectSingleValueFromSchema unless you are going to write rules that use multi-valued LDAP attributes that have a single value. Using config/detectSingleValueFromSchema = true adds the overhead of checking the LDAP schema for each attribute instead of the default behavior (config/detectSingleValueFromSchema = false), which only stores an attribute as multi-valued (in a Collection) if it has more than one value.

This feature also implements changes that allow you to use SUBTREE_SCOPE searches for users and groups. It also allows multiple base userDN and groupDN to be specified. The multiple base DN can be used with SUBTREE_SCOPE searches enabled or disabled.

A SUBTREE_SCOPE search begins at a base userDN (or groupDN) and works down the branches of that base DN until the first user (or group) is found that matches the username (or group name).

To enable SUBTREE_SCOPE searches you must set the Boolean config/objectPropertySubtreeScope env-entry in the ejb-jar.xml for p13n_ejb.jar.jar to true and then you must set the config/userDN and config/groupDN env-entry values to be equal to the base DNs from which you want your SUBTREE_SCOPE searches to begin.

For example, if you have users in ou=PeopleA,ou=People,dc=mycompany,dc=com and in ou=PeopleB,ou=People,dc=mycompany,dc=com then you could set config/userDN to ou=People,dc=mycompany,dc=com and properties for these users would be retrieved from your LDAP server because the user search would start at the “People” ou and work its way down the branches (ou=“PeopleA” and ou=“PeopleB”).

You should not create duplicate users in branches below your base userDN (or duplicate groups below your base groupDN) in your LDAP server. For example, your LDAP server will allow you to create a user with the uid=“userA” under both your PeopleA and your

PeopleB branches. The LdapPropertyManager in p13n_ejb.jar.jar will return property values for the first userA that it finds.

It is recommended that you do not enable this change (by setting config/objectPropertySubtreeScope to true) unless you need the flexibility offered by SUBTREE_SCOPE searches.

An alternative to SUBTREE_SCOPE searches (with or without multiple base DNs) would be to configure multiple base DNs and leave config/objectPropertySubtreeScope set to false. Each base DN would have to be the DN that contains the users (or groups) because searches would not go any lower than the base DN branches. The search would cycle from one base DN to the next until the first matching user (or group) is found.

The new ejb-jar.xml deployment descriptor is fully commented to explain how to set multiple DNs, multiple usernameAttributes (or groupnameAttributes), and how to set the objectPropertySubtreeScope flag.

3. Save and close the file.
4. Replace the original p13n_ejb.jar with the modified version:
   a. Rename the original p13n_ejb.jar to use it as a backup. For example, rename it to p13n_ejb.jar.backup.
   b. JAR the temporary version of p13n_ejb.jar to which you made changes. Name it p13n_ejb.jar.
   c. Copy the new JAR to your application's root directory.
5. Start the server and re-deploy the application.

CHAPTER 4

Overview of Delegated Administration

This section provides an overview of delegated administration in WebLogic Portal. Use the WebLogic Administration Portal to configure delegated administration.

This chapter includes the following sections:

- Using Delegated Administration
- Delegated Administration Role Hierarchy
- Setting Up an Administrative Role

Using Delegated Administration

Delegated administration provides a mechanism for propagating WebLogic Administration Portal privileges within a hierarchy of roles. A Delegated Administration role is a classification of users based on user name, group membership, or by the user's characteristics (or expressions), such as user profile values or time.

In your organization, you might want individuals to have different rights of access to various administration tasks and resources. For example, a system administrator might have access to every feature in the WebLogic Administration Portal. The system administrator might then create a portal administrator role that could manage instances of portal resources in specific desktop views of your portal, and a library administrator role that can manage your portal resource library.

A role policy consists of a role name and role definition. Delegated Administration roles are mapped to administrative functions on portal resources using security policies. Given the

appropriate rights, administrators can delegate both the right to administer a given resource capability and the right for the delegatee to delegate further.

Delegated Administration Role Hierarchy

Roles are dynamic classifications of users who meet specific requirements, such as membership in a group, matching user profile property values, and time of day. A role is used to determine whether to grant or deny access to resources, and to determine which capabilities on those resources are available to the user. The role hierarchy defines the structure for Delegated Administration.

The root Delegated Administration role is defined in the Portal Resource tree as Administrators. Any user mapped to this predefined role has unlimited administrative access in the Administration Portal. Only a user with global administrative rights can change the definition of this root Delegated Administration role.

You have flexibility in the way you set up your administration hierarchy and assign rights to your various administrators. You can create different levels of administrators, each with varying degrees of access. You can also create administrators that can, in turn, delegate administration tasks to other users.

WebLogic Portal includes a default system administrator. The system administrator has unlimited access to administrative tasks anywhere within the enterprise portal application. You can create as many different administrators as you need by creating administrator roles and then assigning specific users, user groups, or user characteristics.

Parent Roles and Child Roles

Delegated Administration roles allow you to determine the portal resources that an administrator can access and what actions administrators can take on those resources. A child role has a subordinate relationship to another role (parent) and is used to determine who can delegate to whom. That is, sub-roles are children in the sense that files are children of directories. A user in a role can delegate only to its sub-roles, providing a way to restrict Delegated Administration.

For example:

RoleA can delegate to:

- Role1
- Role2
- Role3 can delegate to:

  - Role 3.1
  - Role 3.2

RoleB can delegate to:

- Role4
- Role5
- Role6

The user in RoleA cannot delegate to the sub-roles of RoleB as a "peer" role. RoleA can delegate to any of its descendants. Child roles do not inherit the traits of the parent role. If you delete a child role, you are removing it from the system.

Note: When you are establishing your role hierarchy, keep in mind that child roles within a Delegated Administration role must be unique. For example, you cannot have a Delegated Administration role called RoleA with a child role of RoleB if you already have a child role called RoleB elsewhere in the hierarchy.

Setting Up an Administrative Role

You can create Delegated Administration roles at any time. The following process shows all the steps that ensure your administrators are set up correctly:

1. Model your Delegated Administration hierarchy to fit the needs of your organization.
2. Create a Role for each administrator type.
3. Define the role three ways:
   - Add individual users to a role
   - Add a group of users to a role
   - Add users by expressions (attributes)
4. Assign Delegated Administration rights to various resources:
   - Users and groups
   - Portal objects
   - Interaction Management objects
   - Content Management objects

For detailed instructions on setting up delegated administration, see the WebLogic Administration Portal online help at http://edocs.bea.com/wlp/docs81/adminportal/index.html.

CHAPTER 5

Overview of Visitor Entitlements

This section provides an overview of visitor entitlements in WebLogic Portal. Use the WebLogic Administration Portal to configure visitor entitlements.

This chapter includes the following sections:

Using Visitor Entitlements

Visitor Entitlements allow you to define who can access the resources in a portal application and what they may do with those resources. This access is based on the role assigned to a portal visitor, allowing for flexible management of the resources. For example, if you have an Employee Review portlet, you can assign the "Managers" Visitor Entitlement role you created to that portlet, letting only logged in users who belong in that role view the portlet.

A user visiting an application is assigned a role based on an expression that can include their name, the group that they are in, the time of day, or characteristics from their profile. For example, the "gold member" role could be assigned to users because they are part of the frequent flyer program and have flown more than 50,000 miles in the previous year. This role is dynamically assigned to users when they log in to the site.

Setting Up Visitor Entitlements

You can set Visitor Entitlements for a resource by adding one or more roles and setting the capabilities for those roles. The resources that can be entitled within a portal application include

desktops, books, pages, look and feels, and portlets and portlet categories. Each of these has capabilities that are based on the type of resource, as shown in Table 5 -1:

Table 5 -1 Capabilities According to Resource Type

	View	Minimize	Maximize	Edit	Remove
Desktop	x				
Book	x	x	x		
Page	x				
Look & Feel	x				
Portlet Categories	x				
Portlet	x	x	x	x	x

View – The "View" capability determine whether or not the user can see the resource.

Minimize/Maximize – The "Minimize" and "Maximize" capabilities determine whether or not the user is able to minimize or maximize the portlet or book. This applies to books within a page, not to the primary book.

Edit – The "Edit" capability determines whether or not the user can edit the resource's properties.

Remove – The "Remove" capability determines whether or not the user can remove the portlet from a page.

For detailed instructions on setting up visitor entitlements, see the WebLogic Administration Portal online help at http://edocs.bea.com/wlp/docs81/adminportal/index.html.