BEA WebLogic Portal™

Security Guide

Version 4.0
Document Date: December 2001

BEA Security Guide

Document Edition	Date	Software Version
4.0	December 2001	BEA WebLogic Portal 4.0

Contents

About This Document

This document provides information about security in the BEA WebLogic Portal™ product suite, which includes Campaign services, Commerce services, and WebLogic Personalization Server™.

This document covers the following topics:

- Chapter 1, “Introduction,” provides information that may help you to determine your application security requirements early, and explains how WebLogic Portal security relies on J2EE standards and WebLogic Server security. This topic also includes an overview of declarative security with deployment descriptors, an explanation of security behavior in a typical J2EE environment, and some basic information about security realms.
- Chapter 2, “Security Roles and Deployment Descriptors,” describes security roles and deployment descriptors in detail, and provides information about the security roles and deployment descriptor files used in WebLogic Portal. This topic also provides information that will be useful to you when considering security measures in your own applications’ deployment descriptor files.
- Chapter 3, “Security in the WebLogic Portal Sample Applications,” includes security information about each of the sample applications within WebLogic Portal. It describes how the user’s experience might be affected by security measures, and includes more details about back-end security processes such as credit card encryption.
- Chapter 4, “Security in the WebLogic Portal Enterprise JavaBeans,” describes the security restrictions on the methods in each EJB JAR file of WebLogic Portal. This topic is primarily reference material, as it describes the contents of the `weblogic-ejb-jar.xml` and `ejb-jar.xml` deployment descriptors.
- Chapter 5, “Security in the WebLogic Portal Administration Tools and the E-Business Control Center,” provides information on security measures implemented in the WebLogic Portal tools.

- Chapter 6, "Portal Administration and Security," provides additional detail about the three levels of administration that WebLogic Portal supports, and includes information about how the access granted to these administrator users is scoped. Additionally, this topic includes information about how administrative users are managed, and how administration tasks may be delegated. Finally, some information about visitor entitlements for WebLogic Portal is provided.

What You Need to Know

This document is intended mainly for individuals in your organization who act as Application Assmblers/Deployers and System Administrators. See "Development Roles" in the *Strategies for Developing E-Business Web Sites* documentation for more information about these and other organizational roles.

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 Portal 4.0 documentation Home page on the e-docs Web site. A PDF version of this document is also available in the documentation kit on the product CD. Or you can download the documentation kit from the WebLogic Portal portion of the BEA download site. 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 Portal 4.0 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/.

Related Information

The following documents contain additional information about WebLogic Portal and BEA WebLogic Server™ product security, or provide supplemental information that may be helpful for understanding the contents of this *Security Guide*.

- *Deployment Guide*
- *Introduction to WebLogic Security*
- *Programming WebLogic Security*
- *Java 2 Platform Enterprise Edition Specification, v1.3*
- *Enterprise JavaBeans 1.1 Specification*
- *Java Servlet Specification v2.3*

Contact Us!

Your feedback on the BEA WebLogic Portal 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 Portal documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Portal 4.0 release.

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 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.

Convention	Item

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

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

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` *`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 Introduction

This *Security Guide* is designed to help you understand how the WebLogic Portal product suite leverages the security features of the Java™ 2 Platform Enterprise Edition (J2EE) specification and the J2EE-compliant security features of the WebLogic Server platform. It provides information about additional security measures that may have been established for individual application components within the WebLogic Portal product suite, and describes ways that you can modify security settings.

As an introduction to WebLogic Portal security, this topic includes the following sections:

Determining Your Application Security Needs

Security is a critical component of developing e-commerce applications that no organization can afford to ignore. Malicious individuals who gain access to your computer systems can temporarily interrupt business, but those who gain access to your customers' personal data can cause long-term damage to your reputation. Even if you have not had any security mishaps, customers are often hesitant to provide personal information over the Web, which could affect your ability to tailor products and services to customer preferences. Therefore, your organization must develop e-commerce applications that protect customer data and communicate a sense of privacy and purpose through the interface. At the same time, however, the application must not be difficult to navigate or perform slowly because of technical security requirements.

As the previous paragraph suggests, determining your application security needs can be challenging. You must balance the need for securing data with the need to make your site easy to navigate and perform quickly. You may want to ask yourself the following questions when attempting to define your application security requirements:

- *What resources should I protect?*

  There are many resources in the WebLogic Portal suite's applications that can be protected, including Enterprise JavaBeans (EJBs), servlets, and JavaServer Pages (JSPs). Consider the resources you want to protect when deciding the level of security you must provide.

- *From whom am I protecting these resources?*

  Like many e-commerce Web sites, you may consider browsing product catalog pages an acceptable activity for both anonymous and authenticated customers. Once a customer decides to purchase an item, however, you may need to identify that customer and retrieve some of their personal information. This information must be protected from all other customers of the Web site.

  Similarly, the tools included with WebLogic Portal may allow certain administrators to modify the behavior of your e-commerce Web site. Should these administrators be able to access all resources and all data? Or should the most confidential and strategic information be trusted to only a few individuals?

- *What are the consequences of ineffective or failed protection on the resource?*

  In some cases, a fault in your application security is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use the site. Understanding the ramifications associated with unauthorized access to each resource will help you to properly protect it.

As you read the rest of this *Security Guide,* keep your answers to these questions in mind. By thinking upfront about your security requirements, you will be more likely to design security-aware applications, and may be better prepared to execute deployment activities.

Reliance on J2EE Standards and WebLogic Server Security

WebLogic Portal security implementations are based on the security requirements defined in the Java 2 Enterprise Edition (J2EE) specification, as well as the security measures implemented the BEA WebLogic Server^{TM} product.

This dependency has two important implications. First, if implemented properly, you can be confident that the applications you build using WebLogic Portal conform to the latest in J2EE security standards. Second, like the WebLogic Portal applications, your applications will also leverage the centralized security mechanisms of the WebLogic Server container, instead of requiring you to embed security code within individual Enterprise JavaBean (EJB) routines or their clients. Reliance on this centralized security model may significantly reduce the amount of effort required to build and maintain secure applications.

For more information about J2EE security requirements, see the “Security” section in the *Java 2 Platform Enterprise Edition Specification, v1.3*. For more information about how WebLogic Server implements J2EE security, refer to the *Programming WebLogic Security* documentation.

Declarative Security in WebLogic Portal

Because WebLogic Portal relies upon J2EE security requirements and the WebLogic Server's implementation of these requirements, security in the WebLogic Portal applications is primarily declarative.

As stated in the J2EE specification, **declarative security** means specifying an application's security structure—including security roles, access control, and authentication requirements—in a form that is external to the application. This external form is called a **deployment descriptor**, which is a centralized XML document for (among other things) defining protections on application resources. At runtime, the WebLogic Server container uses the deployment descriptor to enforce the specified security restrictions. Access to a protected resource is granted only if a user's security role matches the role declared in the deployment descriptor for that resource. Chapter 2, "Security Roles and Deployment Descriptors," provides more information about deployment descriptors and security roles.

A declarative approach to security has several advantages. First, declarative security is fine-grained, meaning that access can be restricted down to a specific method on a JavaBean. This gives you the flexibility to restrict only those resources that need protection. Second, an Application Assembler/Deployer can map security roles used within an application to the users and groups that already exist for a particular production environment at deployment time. This eliminates the need for individual developers to concern themselves with the security details of particular environments, and allows them to focus on other, possibly more important aspects of application development. Third, because deployment descriptor information is centralized and external to its associated application, it can be changed without modifying any JavaServer Page (JSP) or Enterprise JavaBean (EJB) source code. This architecture may significantly reduce the time developers spend debugging and maintaining code, as well as the likelihood of security holes due to programming errors.

Note: More information about declarative security can be found in the *Enterprise JavaBeans 1.1 Specification*.

Programmatic Security in WebLogic Portal

Programmatic security is also supported by the WebLogic Server container, and WebLogic Portal does occasionally use programmatic security in addition to declarative security. For example, some programmatic security is used in the Portal Administration services because an administrator's association with a portal is determined dynamically, after deployment. The static nature of declarative security (and the fact that it is defined at deployment time) would not work in this case. For more information about Portal Administration, see Chapter 6, "Portal Administration and Security."

Note: For more information about the differences between declarative and programmatic security, see the "Container Based Security" section of the *Java 2 Platform Enterprise Edition Specification, v1.3*.

Security Behavior in a J2EE Environment

The following points provide a high-level overview of how security typically works in a J2EE environment. This information is included to provide some context for the WebLogic Server's security activities, as well as to provide a basic understanding of how an application's deployment descriptors and security roles fit into a security policy.

- *Initial Request*

  A Web client that has not yet been authenticated requests a URL for a protected resource (for example, a JSP). If acting as a Web server, the WebLogic Server detects this and invokes an appropriate authentication mechanism.

- *Authentication*

  The Web server returns a form to the client requesting authentication data (such as a username and password combination). If using basic authentication, the form displayed to the user is part of the browser's default interface. If using form-based authentication, the interface for data collection can be customized by developers in your organization. Once the user enters their authentication information, the client forwards this data back to the Web server, which validates it and sets a credential for the user.

- *Authorization*

  The user's credential (set in the previous step) can now be used in subsequent requests to determine whether that user is authorized to access a particular protected resource. The Web server consults the deployment descriptor associated with the protected resource to determine the security roles that should be granted access. The Web container then tests the user's credential against each security role, looking for user-to-role matches.

- *Request Fulfillment*

  If the Web container obtains an "is authorized" outcome, the Web server returns the result of the original URL request to the client.

  The JSP (or other resource) to which the client now has access may perform a remote method call to an Enterprise JavaBean (EJB), using the user's credential to establish a secure association between the JSP and the EJB. This association is implemented in both the Web container and EJB container. The EJB container enforces access control on EJB methods by consulting the EJB's deployment descriptor to determine which security roles should be granted access to the method. If the EJB container obtains an "is authorized" outcome, the container permits the call to the particular EJB method. The result of the method's execution is returned via the Web server and JSP to the client.

As you can see by this simplified description, the Web server acts as an authentication proxy by gathering authentication data from a client and using it to establish an authenticated connection to a protected resource.

Note: For more detailed information about this process, see the "Security" section in the *Java 2 Platform Enterprise Edition Specification, v1.3*. For more information about authentication and authorization, see "Introduction to WebLogic Server Security" in the *Programming WebLogic Security* documentation.

About Security Realms

As described in "Security Behavior in a J2EE Environment" on page 1-5, a Web server authenticates users and determines which resources within the server users can create, access, or modify. To do this, the Web server uses a security realm. When a user attempts to access a particular resource, the server tries to authenticate and authorize that user by checking the access control list (ACL) and permissions that are assigned to the user within the realm. You can set up multiple security realms, but each instance of a Web server can use only one realm. The server uses the same security realm for your Web site developers and for your customers.

Note: More information about security realms, ACLs, and permissions can be found in the "Introduction to WebLogic Server Security" section of the *Programming WebLogic Security* documentation.

The default security realm for the WebLogic Portal is the RDBMS realm. That is, by default the server stores user IDs and passwords, group definitions, and ACLs in the RDBMS data repository.

WebLogic Portal also provides the following alternate security realms:

- *LDAP Security Realm*

  Provides authentication through a Lightweight Directory Access Protocol (LDAP) server which allows you to manage users in one place, the LDAP directory. When the LDAP security realm is used, the LDAP server authenticates users and groups.

  Note: The LDAP realm is a read-only realm. Therefore, use of the LDAP realm may require additional intervention using third-party tools, especially when performing user, group, and delegated administration management tasks.

- *Windows NT Security Realm*

  Uses Windows NT account information to authenticate users. Users and groups defined through Windows NT can be used by your Web application. You can use the WebLogic Server Administration Console to view this realm, but you must use the facilities provided by Windows NT to define users and groups.

- *UNIX Security Realm*

  Executes a native program, `wlauth`, to authenticate users and groups using UNIX login IDs and passwords. On some UNIX platforms, `wlauth` uses a Pluggable Authentication Module (PAM) that allows you to configure authentication services in a UNIX platform without altering applications that use those services. On UNIX platforms for which PAM is not available, `wlauth` uses the standard login mechanism, including shadow passwords when they are supported. You can use the Administration Console to view this realm, but you must use the facilities provided by the UNIX platform to define users and groups.

- *File Realm*

  When you start the server, the File realm creates user, group, and ACL objects from properties defined through the WebLogic Server Administration Console and stores them in the `fileRealm.properties` file.

  Note: The File realm is designed for use with 10,000 or fewer users. If you have more than 10,000 users, use an alternate security realm.

Implementing a New Custom Realm

As described in "About Security Realms" on page 1-7, WebLogic Portal ships with an implementation of the WebLogic realm that uses a relational database as its backing store. You can, if you would like, replace the out-of-the-box realm with a different realm such as LDAP, or even a custom realm implementation.

If you chose to create your own realm implementation, keep the following points in mind:

- Your realm should implement `weblogic.security.acl.ManageableRealm`.
- If your realm is read-only, it should implement `weblogic.security.acl.ListableRealm`. In this case, the UserManager will not attempt to create or delete users or groups.

  Note: For more information about the UserManager, see "Security Realms and User Profiles" in the *Guide to Building Personalized Applications* documentation.

- In order to support the changing of user passwords, your realm's user object must implement `weblogic.security.acl.CredentialChanger`.
  The single method, `changeCredential`, is used as a callback by the UserManager when `setPassword` is called (either from the WebLogic Portal Administration Tools or from a JSP tag). The first parameter, `oldCredential`, will always be `null`—if you want to confirm a user's current password, this must be done by your application. The second parameter, `newCredential`, will be a String to set as the user's new password.

It is important to note that if a realm that does not implement `ManageableRealm` 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.

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.

For more information about implementing a custom realm, see "Installing a Custom Security Realm" under the topic "Managing Security" in the *WebLogic Server Administration Guide*.

Next Steps

This *Security Guide* is intended mainly for individuals in your organization who act as Application Assemblers/Deployers and System Administrators, and assumes a certain level of familiarity with standard J2EE security implementations and with the security mechanisms of the BEA WebLogic Server.

Note: See "Development Roles" in the *Strategies for Developing E-Business Web Sites* documentation for more information about these and other organizational roles.

If you have no prior or recent experience with J2EE or WebLogic Server, you may want to spend some time learning about these technologies before proceeding. The following documents contain additional information about WebLogic Portal and WebLogic Server product security, or provide supplemental information that may be helpful for understanding the contents of this *Security Guide*.

- *Deployment Guide*
- *Introduction to WebLogic Security*
- *Programming WebLogic Security*
- *Java 2 Platform Enterprise Edition Specification, v1.3*
- *Enterprise JavaBeans 1.1 Specification*
- *Java Servlet Specification v2.3*

If you feel comfortable with J2EE and WebLogic Server security topics, proceed through this guide—either sequentially or nonsequentially—to answer your questions about WebLogic Portal security.

CHAPTER

2 Security Roles and Deployment Descriptors

Security in WebLogic Portal is considered declarative because resource protections are defined in separate configuration files called deployment descriptors, instead of within individual application components.

Because an understanding of security roles is required to understand deployment descriptors, this topic includes conceptual information about security roles, an explanation of how these roles map to BEA WebLogic Server^{TM} principals (users and user groups), and provides information about the security roles used in WebLogic Portal. Next, this topic provides some general information about deployment descriptors, describes the locations of the deployment descriptor files, and introduces some of the security-related elements you will find within these files.

This topic includes the following sections:

- Authorization Using Security Roles
  - What Is a Security Role?
  - WebLogic Portal Security Roles
  - Declaration of Security Roles
  - Users and User Groups as Principals
  - WebLogic Portal User Groups
  - WebLogic Portal Role to Principal Mappings

Authorization Using Security Roles

As described in “Security Behavior in a J2EE Environment” on page 1-5, authentication and authorization mechanisms limit interactions between clients and protected resources. If authentication shows that a user has privileges to a particular resource, a credential is then associated with that user, which authorizes the user to access the same resource during subsequent requests. In a J2EE environment, this authorization mechanism is based on the concept of security roles.

This section includes information on the following:

What Is a Security Role?

A **security role** is an application-specific, logical grouping of users classified by common traits. A security role is a way to assemble users with similar resource access permissions for an application by categorizing them into named collections. For example, perhaps you have some number of users who should be allowed to update just the JSPs for your e-business Web site. It would make sense to group these users together into one security role, named `WebAuthorRole`. This way, you can control the access to the application for all these users through one named security role.

WebLogic Portal Security Roles

The J2EE-compliant security roles defined for the WebLogic Portal applications are:

- `EveryoneRole`—designated for anonymous access.
- `SystemAdminRole`—designated for users who have administration privileges for all aspects of the e-business applications. This includes all areas of the browser-based WebLogic Portal Administration Tools (including Portal, User, Catalog, Order, and Payment Management).
- `DelegatedAdminRole`—designated for users who have specific administration privileges of the e-business applications. Users in this role are not as powerful as those in the `SystemAdminRole` because they are only given access to specific portals (for those further designated as Portal Administrators (PAs)) or group portals (for those further designated as Group Administrators (GAs)).
- `CustomerRole`—designated for customers who are registered with the e-business Web site.
- `RulesReaderRole`—designated as a special role for methods in the Personalization, Campaign, and Portal services that read deployed rule sets. If you wish to call these methods, you must authenticate as a principal in this role.

Declaration of Security Roles

Security roles are declared in each Web application's `web.xml` deployment descriptor file. For more information about deployment descriptors, see "What Is a Deployment Descriptor?" on page 2-7. For more information about the `web.xml` file in particular, see "The web.xml Deployment Descriptors" on page 2-12.

Users and User Groups as Principals

A **user** is similar to an operating system user in that it represents a person. A **user group** is a category of users, classified by common traits such as job title. Categorizing users into user groups makes it easier to control the access permissions for large numbers of users. You can create new users and assign them to user groups in your security realm using the WebLogic Portal Administration Tools, as described in "Creating and Managing Users" in the *Guide to Building Personalized Applications* documentation. For information about the reserved WebLogic Portal user groups, see "WebLogic Portal User Groups" on page 2-5.

Note: Although both user groups and security roles represent categories of users, a security role has a different scope than a user group. Specifically, a security role is scoped only to a *specific application* in the WebLogic Server. A user group is scoped to the *entire* WebLogic Server. In other words, user groups and security realms manage access to resources for an entire WebLogic Server instance, while security roles manage access to resources within a single Web application. (Security roles are described in "What Is a Security Role?" on page 2-3.)

Users and user groups are used as **principals** by application servers like WebLogic Server. The mapping between a principal (used by application servers) and a security role (defined by the J2EE specification) is accomplished in each Web application's `weblogic.xml` deployment descriptor file. (For more information about deployment descriptors or about the `weblogic.xml` file in particular, see "What Is a Deployment Descriptor?" on page 2-7 or "The weblogic.xml Deployment Descriptors" on page 2-13, respectively.)

Mapping a principal (a J2EE user or user group) to a security role confers the defined access permissions to that prinicpal as long as the principal is "in" the role. For example, an application may define a security role called `GuestRole`, which provides read-only access to a small subset of that application's resources. Any principal in the `GuestRole` role would then have read-only access to those resources. Many principals can be mapped to a single J2EE security role. For information about the WebLogic Portal security role to principal mappings, see "WebLogic Portal Role to Principal Mappings" on page 2-5.

WebLogic Portal User Groups

WebLogic Portal uses the following reserved WebLogic Server user groups, which you can configure using the User Management portion of the WebLogic Portal Administration Tools:

- `everyone`
- `SystemAdministrator`—User group for all-powerful administrators.
- `AdminEligible`—Special user group that includes users who are eligible for delegated administration. A user must be placed in this user group before they can be delegated administration tasks.
- `DelegatedAdministrator`—Includes the logical designations of Portal Administrator (PA) and Group Administrator (GA). Users in the `DelegatedAdministrator` user group are in this group for as long as they are either a PA or a GA.
- `wlcs_customer`—User group used exclusively by the Commerce (`wlcs`) Web application.

For more information about the User Management portion of the WebLogic Portal Administration Tools, see "User Management" in the *Getting Started with Portals and Portlets* documentation.

WebLogic Portal Role to Principal Mappings

The security roles defined for the WebLogic Portal Web applications map to the following WebLogic Server users or user groups (principals):

- `EveryoneRole` → `everyone` user group

- `SystemAdminRole` → `SystemAdministrator` user group
- `DelegatedAdminRole` → `DelegatedAdministrator` user group
- `CustomerRole` → `wlcs_customer` user group

Using Role to Principal Mappings To Modify Access At Runtime

In the WebLogic Portal role-to-principal mapping, the `SystemAdminRole` always includes the `SystemAdministrator` user group. Therefore, although you cannot modify access control lists (ACLs) at runtime, you can provide specific users with access at runtime simply by adding them to the appropriate user group. For example, if you want to give user `JohnDoe` access to the WebLogic Portal Administration Tools (which require administrative privileges), simply add `JohnDoe` to the `SystemAdministrator` user group.

Note: For more information about access control lists (ACLs), see “ACLs and Permissions” in the Security Fundamentals topic of the *Programming WebLogic Security* documentation.

Users added to the `DelegatedAdministrator` user group are not automatically made an administrator. They must also be associated with a specific portal or group portal (that is, further designated as a Portal Administrator (PA) or Group Administrator (GA)). For more information, see Chapter 6, “Portal Administration and Security.”

The only time a user must be manually added to the `DelegatedAdministrator` user group is when the LDAP security realm (a non-writable realm) is used. For more information about the LDAP security realm, see “About Security Realms” on page 1-7.

Declarative Security Using Deployment Descriptors

As described in “Declarative Security in WebLogic Portal” on page 1-4, declarative security means specifying an application’s security structure—including security roles, access control, and authentication requirements—in a form that is external to the application. This external form is called a deployment descriptor.

This section includes information on the following:

- What Is a Deployment Descriptor?
- Deployment Descriptor Files and Enterprise Applications
- Location of Deployment Descriptor Files in the Directory Structure
- The web.xml Deployment Descriptors
- The weblogic.xml Deployment Descriptors
- The ejb-jar.xml Deployment Descriptors
- The weblogic-ejb-jar.xml Deployment Descriptors

What Is a Deployment Descriptor?

A **deployment descriptor** is a configuration file with a predefined format that all J2EE-compliant Web applications and Enterprise JavaBeans (EJBs) must use, and that all J2EE-compliant servers (such as the BEA WebLogic Server) must know how to read. This format is specified in an XML Document Type Definition, or DTD, and thus has a `.xml` extension. As its name implies, the deployment descriptor describes various deployment settings including servlets, security roles and secured resources, JSP deployment options, and other properties of an application. For more detailed information about deployment descriptors, see the *Java 2 Platform Enterprise Edition Specification, v1.3*.

Deployment Descriptor Files and Enterprise Applications

A J2EE application, which may or may not be saved as a compressed Enterprise ARchive (EAR) file, generally consists of:

- One or more Web applications, which may or may not be saved as compressed Web ARchive (WAR) files.
- One or more Enterprise JavaBeans (EJBs) that provide e-business functions, saved as compressed Java ARchive (JAR) files.
- Other resources or configurations (such as security settings in deployment descriptors) that are shared among application modules.
- Special Web applications that deploy data and administer your Web applications.

Thus, although deployment descriptors provide a central location for security-related deployment information, an enterprise application typically requires more than one deployment descriptor file to communicate its security requirements to the server. For each enterprise application, you will need to have:

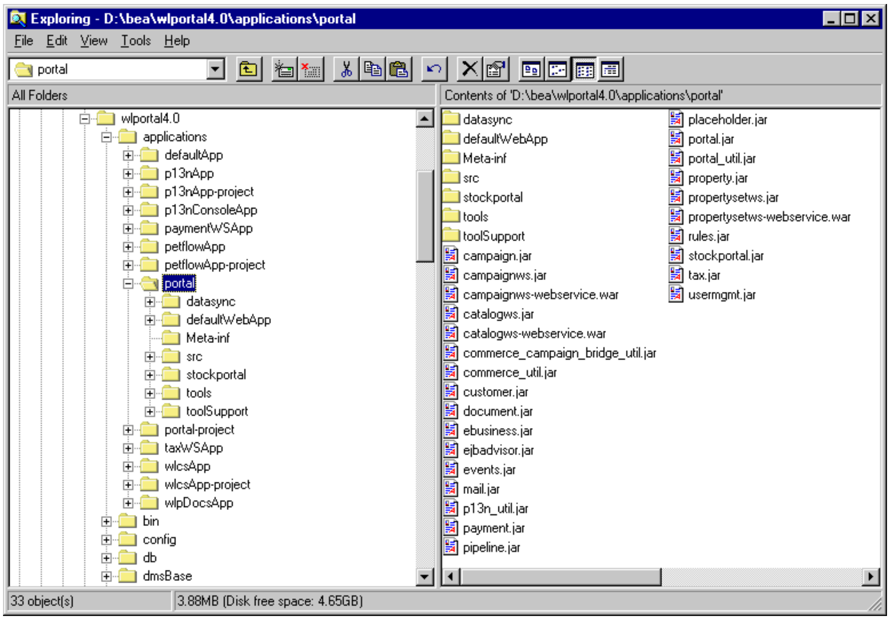
- An `application.xml` deployment descriptor, located in the enterprise application's `META-INF` directory. The `application.xml` deployment descriptor contains all J2EE security roles that constituent Web applications use. For more information, see "Declarations of Security Roles" in the "Modify application.xml" section of the *Deployment Guide*.
- For each Web application that comprises the enterprise application, there are two deployment descriptors: `web.xml` and `weblogic.xml`. These deployment descriptors are discussed in "The web.xml Deployment Descriptors" on page 2-12 and "The weblogic.xml Deployment Descriptors" on page 2-13.
- For each EJB JAR that comprises the enterprise application, there are two deployment descriptors: `ejb-jar.xml` and `weblogic-ejb-jar.xml`. These deployment descriptors are discussed in "The ejb-jar.xml Deployment Descriptors" on page 2-13, and "The weblogic-ejb-jar.xml Deployment Descriptors" on page 2-15.

Location of Deployment Descriptor Files in the Directory Structure

WebLogic Portal is a collection of prewritten Web applications and Enterprise JavaBeans (EJBs), organized into various enterprise applications. These enterprise applications are located in the `PORTAL_HOME\applications` directory, where `PORTAL_HOME` is the directory in which you installed WebLogic Portal. An example of an enterprise application is the `portal` application, which can be found at `PORTAL_HOME\applications\portal`.

The root directory for each enterprise application may contain JAR files for the Enterprise JavaBeans (EJBs) that comprise the enterprise application. Figure 2-1 illustrates this, using the `portal` enterprise application as an example.

Figure 2-1 EJB JAR Files in the WebLogic Portal Directory Structure


Exploring - D:\bea\wlportal4.0\applications\portal
File Edit View Tools Help
portal
All Folders
Contents of 'D:\bea\wlportal4.0\applications\portal'

```
wlportal4.0
  applications
    defaultApp
    p13nApp
    p13nApp-project
    p13nConsoleApp
    paymentWSApp
    petflowApp
    petflowApp-project
    portal
      datasync
      defaultWebApp
      Meta-inf
      src
      stockportal
      tools
      toolSupport
    portal-project
    taxWSApp
    wlcsApp
    wlcsApp-project
    wlpDocsApp
  bin
  config
  db
  dmsBase
```

```
datasync                              placeholder.jar
defaultWebApp                         portal.jar
Meta-inf                              portal_util.jar
src                                   property.jar
stockportal                           propertysetws.jar
tools                                 propertysetws-webservice.war
toolSupport                           rules.jar
campaign.jar                          stockportal.jar
campaignws.jar                        tax.jar
campaignws-webservice.war             usermgmt.jar
catalogws.jar
catalogws-webservice.war
commerce_campaign_bridge_util.jar
commerce_util.jar
customer.jar
document.jar
ebusiness.jar
ejbadvisor.jar
events.jar
mail.jar
p13n_util.jar
payment.jar
pipeline.jar
```

```
33 object(s)    3.88MB (Disk free space: 4.65GB)
```

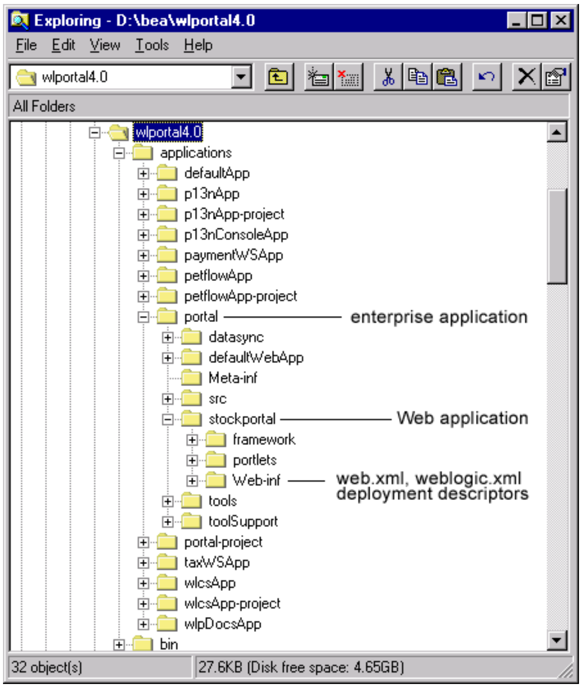
It is within each JAR file that the `ejb-jar.xml` and `weblogic-ejb-jar.xml` deployment descriptors can be found. These deployment descriptors are discussed in "The ejb-jar.xml Deployment Descriptors" on page 2-13, and "The weblogic-ejb-jar.xml Deployment Descriptors" on page 2-15. The contents of the `ejb-jar.xml` and `weblogic-ejb-jar.xml` files for each of the prewritten EJBs in WebLogic Portal are also described in Chapter 4, "Security in the WebLogic Portal Enterprise JavaBeans."

Beneath each enterprise application's root directory, there are two important subdirectories: `META-INF` and `<webapp_name>\WEB-INF`.

- The `META-INF` subdirectory contains the `application.xml` deployment descriptor file for the enterprise application. For more information, see "Declarations of Security Roles" in the "Modify application.xml" section of the *Deployment Guide*.
- The individual Web applications that comprise each enterprise application are located in subdirectories beneath the enterprise applications. For example, the `stockportal` Web application is located in the `PORTAL_HOME\applications\`

`portal\stockportal` directory. A portion of this directory structure is shown in Figure 2-2.

Figure 2-2 Location of Web Application Deployment Descriptors in the WebLogic Portal Directory Structure



Within the Web application's root directory, there is a `WEB-INF` subdirectory. The `WEB-INF` directory contains the two deployment descriptors for the Web application: `web.xml` and `weblogic.xml`. These deployment descriptors are discussed in "The web.xml Deployment Descriptors" on page 2-12 and "The weblogic.xml Deployment Descriptors" on page 2-13.

The web.xml Deployment Descriptors

Along with other information, the `web.xml` deployment descriptor may contain several sets of XML elements for implementing the J2EE declarative security model. Each `web.xml` deployment descriptor may contain XML elements for the following security-related topics:

- *Listen Ports for Webflow-Generated URLs*

  These `<context-param>` elements enable Webflow to encode port numbers in URLs that are different from the port numbers that the WebLogic Server uses.

- *Declarations of Secure Links*

  These `<context-param>` elements declare a set of files, Pipelines, and Input Processors that need to be accessed via HTTPS.

- *Session Configuration*

  The `<session-config>` element defines the session parameters for the Web application, including session timeout values.

- *Declarations of Secure JSPs*

  The `<security-constraint>` XML element declares a collection of resources (JSPs) that only specific roles can access.

- *Login Configuration*

  The `<login-config>` element determines how users are authenticated.

- *Declaration of Security Roles*

  You must include a `<security-role>` element to declare all of the roles that you use in the `<security-constraint>`.

These topics are described in more detail in the the “Review and Modify web.xml” section of the *Deployment Guide*. More information can also be found in “web.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* documentation.

The weblogic.xml Deployment Descriptors

In addition to the `web.xml` deployment descriptor, each Web application also requires a `weblogic.xml` deployment descriptor, which declares deployment properties specific to the WebLogic Server. Along with other information, each `weblogic.xml` deployment descriptor may contain XML elements for the following security-related topics:

- *Security-Role Mappings*

  The `<security-role-assignment>` elements map J2EE security roles to users or groups that a security realm uses to define ACLs.

- *Check Authentication on JSP Forwarding*

  The `<container-descriptor>` element requires authentication of any forwarded requests from a servlet or JSP.

- *Cookies*

  The `<session-param>` elements determine whether the default cookie name for all Web applications on the current server instance is used, and allows you to set the cookie domain for the application.

- *Customer Profile Initialization*

  The value of the `<auth-filter>` element specifies whether customer profile information is initialized when a customer logs in using a form.

These topics are described in more detail in the the “Modify weblogic.xml” section of the *Deployment Guide*. More information can also be found in “weblogic.xml Deployment Descriptor Elements” in the *WebLogic Server Assembling and Configuring Web Applications* documentation.

The ejb-jar.xml Deployment Descriptors

An `ejb-jar.xml` file is the primary deployment descriptor for an Enterprise JavaBean (EJB). It is in this XML file that an enterprise application’s individual JavaBeans are registered with appropriate security constraints. As such, the root element of this deployment descriptor element is `<ejb-jar>`. The `<ejb-jar>` element contains mandatory structural information about all included enterprise beans (defined in the

<enterprise-beans> subelement), and may include an application-assembly descriptor. If present, the assembly descriptor contains the enterprise bean's security configuration information (in the <assembly-descriptor> subelement). This element structure is shown in Listing 2-1.

Listing 2-1 ejb-jar.xml XML Element Structure

```
<ejb-jar>

   <enterprise-beans>

       <entity>Declares an entity bean
          <security-role-ref>Declares the security role for the
          entity bean</security-role-ref>
       </entity>

       <session>Declares a session bean
          <session-type>Stateful or Stateless</session-type>
          <security-role-ref>Declares the security role for the
          session bean</security-role-ref>
       </session>

   </enterprise-beans>

   <assembly-descriptor>

       <security-role>
          <description>Documentation of the security
          role</description>
          <role-name>Name of the security role</role-name>
       </security-role>

       <method-permissions>

          <role-name>Name of the security role</role-name>

          <method>
              <ejb-name>Name of the bean</ejb-name>
              <method-name>Name of the bean's method</method-name>
          </method>

       </method-permissions>

   </assembly-descriptor>

</ejb-jar>
```

As Listing 2-1 also shows, the `<enterprise-beans>` element contains `<entity>` subelements when declaring entity beans, or `<session>` subelements when declaring session beans. The `<entity>` and `<session>` elements may further contain `<security-role-ref>` subelements, which enable the EJB to do programmatic security checking (if such behavior is desired). Also within the `<session>` element is a `<session-type>` subelement that describes whether the session bean is stateful or stateless.

Note: For more information about the differences between declarative and programmatic security, see "Container Based Security" in the *Java 2 Platform Enterprise Edition Specification, v1.3*.

If included, the `<assembly-descriptor>` element includes security roles and the individual method permissions associated with these security roles. The `<security-role>` subelements declare all the roles by which bean method calls will be authorized. Each security role requires a mapping to an actual group name in the corresponding `weblogic-ejb-jar.xml` file. (The `weblogic-ejb-jar.xml` file is described in the following section.)

The assembly descriptor's `<method-permission>` subelement declares authorizations for each method in a bean. These entries are typically listed by security role (`<role-name>`), then by bean (`<ejb-name>`) and then by method (`<method-name>`). Therefore, there is usually one `<method-permission>` entry for each security role defined in the previously described `<security-role>` element.

Note: Because WebLogic Portal consists of many EJB JAR files, the detailed contents of the `ejb-jar.xml` and `weblogic-ejb-jar.xml` deployment descriptors can be found in Chapter 4, "Security in the WebLogic Portal Enterprise JavaBeans."

The weblogic-ejb-jar.xml Deployment Descriptors

One purpose of the `weblogic-ejb-jar.xml` deployment descriptor is to associate the security role names that may be needed for a particular service (as defined in the corresponding `ejb-jar.xml` file) to principal names. Listing 2-2 shows the syntax of the `<security-role assignment>` element.

Listing 2-2 Syntax of the Security-Role Assignment Element

```
<security-role-assignment>

   <role-name>Name of security role</role-name>
   <principal-name>Corresponding principal name</principal-name>

</security-role-assignment>
```

Notes: The role to principal mappings used in the `web.xml` deployment descriptor may also be used in `weblogic-ejb-jar.xml`. For more information about the `web.xml` application deployment descriptor, see "The web.xml Deployment Descriptors" on page 2-12.

For more information about security roles and principals, see "Users and User Groups as Principals" on page 2-4.

CHAPTER

3 Security in the WebLogic Portal Sample Applications

Out-of-the-box, WebLogic Portal includes several sample applications that demonstrate various capabilities of the product, and some that may provide you with a starting point for building your own e-commerce applications. These sample applications must occasionally implement J2EE security measures. For example, because the JavaServer Page (JSP) templates included in the Commerce (`wlcs`) Web application simulate a real-world storefront (including a check out process), some of the templates must be protected.

This topic describes how a visitor or customer's user experience might be affected by security in the sample applications, explains the back-end processes that support security in these applications, and describes the contents of each application's `web.xml` and `weblogic.xml` deployment descriptors. It also provides details about credit card security measures.

This topic includes the following sections:

- Security in the Commerce (wlcs) Web Application
  - Accessing the Commerce (wlcs) Web Application
  - Protected JavaServer Page (JSP) Templates
  - Logging In with Form-Based Authentication
  - Session Inactivity
  - SSL and Declarative Transport

- Credit Card Security Service
- Contents of the Commerce (wlcs) Web Application's Deployment Descriptors

■ Security in the Stock Portal Web Application

- Accessing the Stock Portal Web Application
- Anonymous Versus Authenticated Portal Visitors
- Logging Into the Portal
- Webflow and Portal Security
- Contents of the Stock Portal Web Application's Deployment Descriptors

Security in the Commerce (wlcs) Web Application

The Commerce (`wlcs`) Web application includes a set of JavaServer Page (JSP) templates that provide a model for your home page, search and browse pages, shopping cart pages, and many other pages that are commonly used on e-commerce sites. Web content developers can use the JSP templates and add customized HTML tags to match your organization's branding requirements, design preferences, navigation options, and the content of your product catalog. In summary, the JSP templates are a great starting point for building your own e-commerce Web application!

Because some e-commerce Web pages display or allow registered customers to modify personal information (such as billing information or demographic data), the included JSP templates that serve these purposes must be protected. This section describes which JSP templates are protected, how users can access the templates, and the techniques used to implement these security measures.

Accessing the Commerce (wlcs) Web Application

On Windows systems, you can view the Commerce Web application by first starting the WebLogic Portal server and then selecting Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Portal 4.0 → Commerce Templates.

The default username/password combination for the `wlcs` templates is `democustomer` and `password`.

The files for the Commerce (`wlcs`) Web application are located in `PORTAL_HOME\applications\wlcsApp\wlcs` directory, where `PORTAL_HOME` is the directory in which you installed the product.

Protected JavaServer Page (JSP) Templates

Unless a Web page contains or requests sensitive customer information, there is no reason to prevent any user from viewing that page. For example, a potential customer who visits your e-commerce site for the first time should be able to browse through your product catalog with ease. Typical browsing interactions (excluding campaigns) should not require this user to register or log into your Web site.

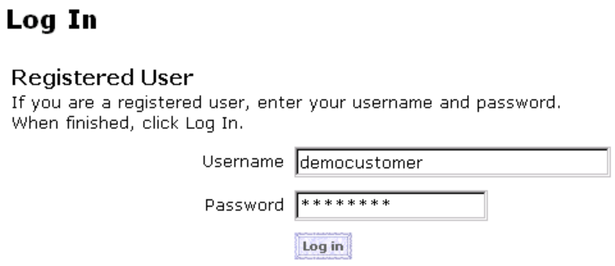
With this concept in mind, the Commerce (`wlcs`) Web application allows anonymous users to move through the JSP templates freely, until they request a page that is protected (that is, contains or requests sensitive information). Protected pages are those located below the `commerce\order` and `commerce\user` directories.

Logging In with Form-Based Authentication

The `main.jsp` template is the default home page for the Commerce (`wlcs`) Web application, and can be accessed by anonymous users. Anonymous users can browse the store or perform other activities (search, view cart, and so on) using the hyperlinks presented on the `main.jsp` template. Essentially, anonymous users can access all of the product catalog JSP templates, which are described in detail in the “Product Catalog JSP Templates” topic of the *Guide to Building a Product Catalog* documentation. (In this documentation you will also find a detailed explanation of the `main.jsp` template.)

Although a Log In link is provided on the `main.jsp` template, users are not immediately required to log in. Only when an anonymous user attempts to access a protected Web page (or explicitly clicks the Log In link) are they directed to the `login.jsp` template for a username and password (see Figure 3-1). Users who choose to log in prior to accessing protected pages and are already authenticated are not prompted to log in again unless they log out or their session times out. (For more information about session timeouts, see "Session Inactivity" on page 3-5.)

Figure 3-1 Portion of the login.jsp Template


Log In

Registered User

If you are a registered user, enter your username and password.
When finished, click Log In.

Username democustomer

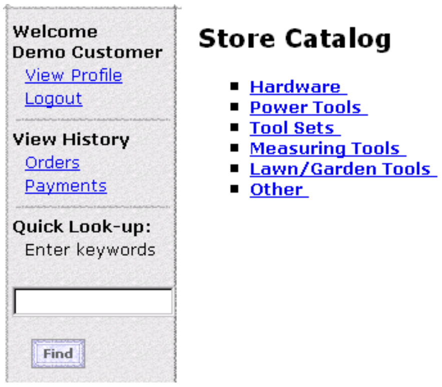
Password ********

The Commerce (`wlcs`) Web application uses form-based authentication techniques to present the `login.jsp` template to the user. Form-based authentication allows developers to customize the authentication user interface presented by an HTTP browser. In other words, you can specify the HTML form or JSP file that prompts for the username and password. Or, you can modify the look and feel of the `login.jsp` template to adhere to your organization's marketing requirements.

When the user submits their username and password to gain access to their account, the submitted information is passed to the WebLogic Server for verification. The security mechanism in WebLogic Server verifies that the user is part of the `wlcs_customer` group and that the user has supplied the correct password

If the supplied username/password combination is valid and the user arrived at the `login.jsp` template on their way to a protected page, the user is logged in and automatically directed to the protected JSP they were attempting to access. If the username/password combination was validated after explicitly requesting `login.jsp` from the `main.jsp` template, the application loads the secured version of the `main.jsp` template (`secureMain.jsp`). The user is now considered a registered user, so this page welcomes the user with a personalized greeting and presents the user with additional links in the left navigation bar (see Figure 3-2).

Figure 3-2 Portion of main.jsp Template After User Login



In either case, if the supplied username/password combination is deemed invalid, the `badlogin.jsp` template is displayed. The `badlogin.jsp` template is essentially the same as the `login.jsp` template, but includes an appropriate error message.

Notes: The `login.jsp` and `badlogin.jsp` templates are described in detail in the “Customer Registration and Login Services” topic of the *Guide to Registering Customers and Managing Customer Services* documentation.

While you are logging into the Commerce (`wlcs`) Web application, your browser may display information related to the demonstration certificate in a pop-up window. If you use the `wlcs` sample application as a basis for your own e-commerce Web site, be sure to purchase and install a real server license from a certificate authority. For more information about certificates, see “Digital Certificates” in the *Programming WebLogic Server Security* documentation.

Session Inactivity

If an authenticated user does not interact with the templates for a period of 15 minutes, the session will be deemed inactive. If the user then returns to the templates and attempts to access a protected page, the `sessiontimeout.jsp` template will be displayed. This page informs the user that because of the inactivity, they have automatically been logged out of the system and will need to log in again. This automatic logout prevents unauthorized individuals from accessing a logged-in user's account information if the user were to leave their machine for a certain period of time.

Note: The session timeout interval is set in the application's `web.xml` deployment descriptor, and can be customized to something other than the default value of 15 minutes. For more information, see "Contents of the Commerce (wlcs) Web Application's Deployment Descriptors" on page 3-14.

SSL and Declarative Transport

The Webflow and Pipeline mechanisms that direct the presentation and business logic associated with the Commerce JSP templates make use of SSL and declarative transport mechanisms. Links that invoke protected JSP files, as well as certain Input Processors and Pipelines, need to be accessed via the HTTPS protocol. There are a number of these links already defined in the Commerce (`wlcs`) Web application's `web.xml` deployment descriptor. Secured JSP templates that rely on SSL also require a setting in the `web.xml` file that indicates the transport guarantee. This guarantee can be `CONFIDENTIAL` or `INTEGRAL`. A `CONFIDENTIAL` setting prevents other entities from observing the contents of the transmission, while an `INTEGRAL` setting prevents the data from being changed while transmitting between the client and server. For more information on these settings for the Commerce (`wlcs`) Web application, see "Contents of the Commerce (wlcs) Web Application's Deployment Descriptors" on page 3-14.

Credit Card Security Service

All credit card information your customers provide is considered sensitive and is encrypted for security purposes. This information is decrypted only when absolutely necessary during specific payment processing activities (such as authorization). On the order confirmation JSP template (`confirmorder.jsp`), for example, only the last 4 digits of a customer's credit card are displayed.

This section provides detailed information about the credit card security service, including:

- Encryption/Decryption Implementation

- Customizable Security Settings
- Methods for Supplying the Private Key Encryption Password

Encryption/Decryption Implementation

The Commerce services encryption mechanism relies on libraries from the WebLogic Server platform and is based upon RSA's public key infrastructure. A public key is used to encrypt a customer's credit card information, while a private key is used to decrypt it when required.

Note: For more information about RSA security standards, see http://www.rsasecurity.com/.

The public key is stored in the database for use by the `EncryptCreditCardPC` Pipeline Component, while the private key is itself encrypted using a password you supply, and stored in the database. When invoked from the Webflow, the `EncryptCreditCardPC` Pipeline Component reads the customer-provided credit card information from the Pipeline Session, encrypts it using the public key, and then places it back into the Pipeline Session. This encrypted data is subsequently persisted to the database. Decryption is accomplished using a back-end component and the private key. Again, decryption is initiated only in stages of the ordering process where this data is absolutely necessary.

Note: For more technical information about the Credit Card Security Service, please contact your BEA representative.

Customizable Security Settings

Although the Commerce services specify default settings for the Credit Card Security Service, you can customize them. The security settings reside in the `weblogiccommerce.properties` file (located in the `PORTAL_HOME` directory, where `PORTAL_HOME` is the directory in which you installed WebLogic Portal). These security settings are shown in Listing 3-1.

Listing 3-1 Security Settings in weblogiccommerce.properties

```
####################################################
# Properties required for the Security Services
####################################################
```

```
# Credit card encryption services are turned on by setting this
# property to true. Commenting out the property or setting it to
# false will disable credit card encryption.

is.encryption.enabled=true


# The name of the security table and column names for the public
# and private encryption keys can be specified using the properties
# below.

security.table.name=WLCS_SECURITY
security.backup.table=WLCS_SECURITY_BACKUP
public.key.column.name=PUBLIC_KEY
private.key.column.name=PRIVATE_KEY


# The key bit size desired. Key bit length and length of data that
# can be encrypted are related # as follows:

# KEY BIT LENGTH(bits)          DATA LENGTH (bytes)
#      512                      53
#      1024                     117
#      2048 (MAX LENGTH)        245

key.bit.size=1024


# This optional parameter specifies the private key password used
# to decrypt the encrypted credit card encryption private key.
# WARNING: Setting this property will start up the server without
# prompting for a password.

private.key.password=WLCS
```

First, the `is.encryption.enabled` property enables encryption mechanisms. Please note that a value of `false` (or no value at all) will disable encryption mechanisms. BEA has assigned this property a default value of `true`.

Next is a series of properties that allow you to specify the names of the security tables (primary and backup) and the columns in which the public and private keys will be stored. BEA has assigned default values to these properties, but you can modify them based on your database.

Following the properties related to the database, the `key.bit.size` property allows you to specify the encryption key length. By default, the Commerce services ship with a 1024-bit key size, although you may want to modify the encryption strength by changing the key size. Table 3-1 illustrates the possible key bit values.

Table 3-1 Key Bit Values

Length (Bits)	Data Length (Bytes)
512	53
1024	117
2048	245

Note: The higher the key size, the stronger the encryption and more secure the data. However, keep in mind that there is a trade-off in increasing the size of the key—your data may be more secure, but it may take longer to encrypt the data.

Lastly, the `private.key.password` property allows you to specify, in the `weblogiccommerce.properties` file, the password used to encrypt the private key. Please note that BEA does not recommend use of this property. Rather, the private key should be supplied by an administrator during server startup. For more information about supplying the private key, see "Methods for Supplying the Private Key Encryption Password" on page 3-9.

Note: If not used, the `private.key.password` property should be commented out with a # symbol. BEA has assigned this property a default value of `WLCS`, but this is for demonstration purposes only.

Methods for Supplying the Private Key Encryption Password

As previously mentioned, the private key used to encrypt customer credit cards is itself encrypted with a password before being stored in the database. There are three methods by which you can supply this password:

- Specify the password in the `weblogiccommerce.properties` file, which will be read by a startup class (not recommended).
- Specify the password at server startup using the console (recommended).

- Specify the password after server startup using a secure Web form (recommended).

Specifying the Password in weblogiccommerce.properties (Default)

The first method for specifying the private key encryption password is to specify the password as a value for a property in the `weblogiccommerce.properties` file.

Note: BEA does not recommend this method because by providing the password in a simple text file, you leave yourself vulnerable to security attacks. Anyone who gains access to this file can read the password you use to encrypt the private key, and thus gain access to it.

To use this method, follow these steps:

1. In the `weblogiccommerce.properties` file (located in the `PORTAL_HOME` directory, where `PORTAL_HOME` is the directory in which you installed the product), use the `private.key.password` property to specify the password.
2. Set up the class `com.beasys.commerce.ebusiness.security.KeyBootstrap` as a startup class in the WebLogic Server console. For more information on startup classes, see "Registering Startup and Shutdown Classes" in the *WebLogic Server Administration Guide*.

   Note: The `KeyBootstrap` class is located in the `PORTAL_HOME\lib\commerce_system.jar` file, and must be in the system classpath when starting WebLogic Portal. This class is already registered with the `portalDomain` and the `wlcsDomain`.

Specifying the Password at Server Startup Using the Console

The second method for specifying the private key encryption password is for an administrator to specify the password at server startup using the server console.

To use this method, follow these steps:

1. In the `weblogiccommerce.properties` file (located in the `PORTAL_HOME` directory, where `PORTAL_HOME` is the directory in which you installed the product), comment out the `private.key.password` property line with a # symbol.

2. In the WebLogic Server console, ensure that the `com.beasys.commerce.ebusiness.security.KeyBootstrap` class is not a startup class. For more information on startup classes, see "Registering Startup and Shutdown Classes" in the *WebLogic Server Administration Guide*.

   Note: The `KeyBootstrap` class is located in the `PORTAL_HOME\lib\commerce_system.jar` file, and must be in the system classpath when starting WebLogic Portal. This class is already registered with the `portalDomain` and the `wlcsDomain`.

Specifying the Password After Server Startup Using a Secure Web Form

The third method for specifying the private key encryption password allows an administrator to enter the password on a secure Web form, so the password is stored in memory on your system instead of in a text file.

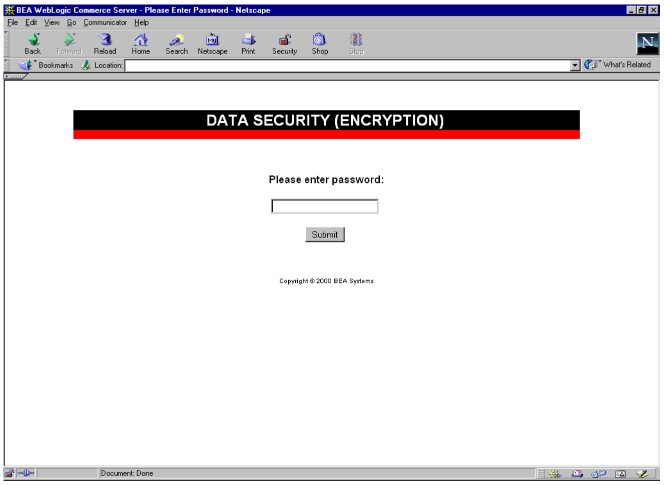
To use this method, follow these steps:

1. In the WebLogic Server console, ensure that the `com.beasys.commerce.ebusiness.security.KeyBootstrap` class is not a startup class. For more information on startup classes, see "Registering Startup and Shutdown Classes" in the *WebLogic Server Administration Guide*.

   Note: The `KeyBootstrap` class is located in the `PORTAL_HOME\lib\commerce_system.jar` file, and must be in the system classpath when starting WebLogic Portal. This class is already registered with the `portalDomain` and the `wlcsDomain`.

2. Point your Web browser to `<hostname>:port/tools/security/security_getPassword.html`, to load the secure Web form shown in Figure 3-3.

Figure 3-3 security_getPassword.html


BEA WebLogic Commerce Server - Please Enter Password - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: What's Related
DATA SECURITY (ENCRYPTION)
Please enter password:
Submit
Copyright © 2000 BEA Systems
Document: Done

3. Specify the private key encryption password in the form field and click the Submit button.

   On submission, this page will invoke the `EncryptionServlet` and `KeyGeneratorServlet` registered in the `web.xml` file (located in the `PORTAL_HOME\applications\wlcsApp\tools\Web-inf` directory, where `PORTAL_HOME` is the directory in which you installed the product).

Important Notes About Supplying Your Password

You must supply the password for all nodes in a cluster. Should one node in the cluster fail, other machines that know the private key encryption password can be used for failover.

The first time you enter the password, you will be asked to confirm whether or not you want to generate new keys. If this is indeed the first time you are entering the password, you do want to generate new keys. However, be sure to select a password that is memorable. All credit cards accepted by your site will be encrypted using this password, and cannot be decrypted if you forget your password.

If you are asked to confirm whether or not you want to generate new keys and you are using the same password, then the keys cannot be found in the database. If no data was encrypted using the old keys, you can regenerate the keys. However, if data has already been encrypted using the old keys, this data will be lost because it cannot be read using the new keys. If you have data encrypted with the old keys, you should stop the server, check the database, and verify the properties in the `weblogiccommerce.properties` file to ensure that the system is set up properly.

During server startup, any orders placed before the password is entered will be persisted with a payment transaction in the `RETRY` state. After supplying the password, administrators should use the Payment Management WebLogic Portal Administration Tools to reauthorize the transaction. For more information about using the Payment Management Administration Tool, see "Using the Order and Payment Management Pages" in the *Guide to Managing Purchases and Processing Orders* documentation.

What if I Want to Change My Password?

Because all the credit cards that have been encrypted use the private key encryption password, it is not recommended that you change this password. However, there may be the rare occasion (for example, if the password has been compromised) when you will need to change the password. Changing the password means changing the public and private key pair. Therefore, you must follow this process when changing the password:

- Use the old password (and thus the old key pair) to decrypt old credit card numbers. The credit card numbers will now be in plain text. Store the credit card numbers in a data structure that preserves the original organization.
- Create a new key pair using a new password.
- Using the new key pair, re-encrypt the plain text credit card numbers from the data structure.

Note: Changing the password is especially difficult if you have a lot of encrypted data. Again, this process is not recommended and should not be done unless absolutely required.

Contents of the Commerce (wlcs) Web Application's Deployment Descriptors

This section provides detailed information about the security-related XML elements in the Commerce (`wlcs`) Web application's deployment descriptor files. It is intended as a reference. For background conceptual information about the security elements listed in this section, see "The web.xml Deployment Descriptors" on page 2-12 and "The weblogic.xml Deployment Descriptors" on page 2-13.

web.xml

Along with other information, the `web.xml` deployment descriptor for the Commerce (`wlcs`) Web application contains several sets of XML elements for implementing the J2EE declarative security model. The contents of these elements are shown in the following code listings:

- Listen Ports for Webflow-Generated URLs
- Declarations of Secure JSPs
- Login Configuration
- Declaration of Security Roles

Listing 3-2 Listen Ports for Webflow-Generated URLs

```
<context-param>
   <param-name>HTTP_PORT</param-name>
  <param-value>7501</param-value>
</context-param>

<context-param>
  <param-name>HTTPS_PORT</param-name>
  <param-value>7502</param-value>
</context-param>
```

Listing 3-3 Declarations of Secure JSPs

```
<security-constraint>
   <web-resource-collection>

       <web-resource-name>Administration Tool Pages
       </web-resource-name>
       <description>The Administration Tool Pages</description>
       <url-pattern>/tools/*</url-pattern>
       <url-pattern>/repository/*</url-pattern>
       <url-pattern>/security/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
   </web-resource-collection>

   <auth-constraint>
       <description>Administrators</description>
       <role-name>SystemAdminRole</role-name>
   </auth-constraint>

   <user-data-constraint>
       <transport-guarantee>NONE</transport-guarantee>
   </user-data-constraint>

</security-constraint>
```

Listing 3-4 Login Configuration

```
<login-config>
   <auth-method>BASIC</auth-method>
</login-config>
```

Listing 3-5 Declaration of Security Roles

```
<security-role>
   <description>Administrators</description>
   <role-name>SystemAdminRole</role-name>
</security-role>
```

weblogic.xml

Along with other information, the `weblogic.xml` deployment descriptor for the Commerce (`wlcs`) Web application contains several sets of XML elements for implementing the J2EE declarative security model. The contents of these elements are shown in the following code listings:

- Security-Role Mappings
- Check Authentication on JSP Forwarding
- Cookies

Listing 3-6 Security-Role Mappings

```
<security-role-assignment>
   <role-name>SystemAdminRole</role-name>
   <principal-name>SystemAdministrator</principal-name>
</security-role-assignment>
```

Listing 3-7 Check Authentication on JSP Forwarding

```
<container-descriptor>
   <check-auth-on-forward />
</container-descriptor>
```

Listing 3-8 Cookies

```
<session-descriptor>
   <session-param>
       <param-name>CookieName</param-name>
       <param-value>JSESSIONID_WLCS_TOOLS</param-value>
   </session-param>

   .
   .
   .
</session-descriptor>
```

Security in the Stock Portal Web Application

The Stock Portal Web application includes a set of JavaServer Pages (JSPs) that illustrate an Internet portal. Internet portals are a key part of many e-commerce applications; they provide an entry point to the Internet as well as value-added services such as searching and application integration. Within portal pages are a number of portlets, or highly focused channels of information. 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. Among other things, portal technology allows Web site visitors to select and arrange the portlets that appear within their portals, allowing for truly personalized information display.

Visitors that wish to customize the portal by selecting pages, the layouts or portlets that appear within those pages, or apply portlet skins to change the look of their portal must first log in so that their preferences can be saved. This section describes which JSP templates are used for the registration process, how users can access the templates, and the techniques used to implement these security measures.

Accessing the Stock Portal Web Application

On Windows systems, you can view the Stock Portal Web application by first starting the WebLogic Portal server and then selecting Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Portal 4.0 → WebLogic Portal Example.

The default username/password combination for the Stock Portal Web application is `administrator` and `password`.

The files for the Stock Portal Web application are located in `PORTAL_HOME\applications\portal\stockportal` directory, where `PORTAL_HOME` is the directory in which you installed the product.

Anonymous Versus Authenticated Portal Visitors

There is no reason to prevent any anonymous visitor from browsing the Stock Portal Web application. The `portal.jsp` template is the default home page for the Stock Portal Web application; it allows all anonymous visitors of the portal to access and browse it. However, anonymous visitors are presented with a default portal that contains portlets, color schemes, organization, content, and so on that are not currently customized to the visitor's personal preferences. For the portal to display information based on visitor preferences, the Stock Portal Web application requires that visitors first be authenticated by logging into the portal. Only after a successful authentication can visitors request a personalized version of their portal.

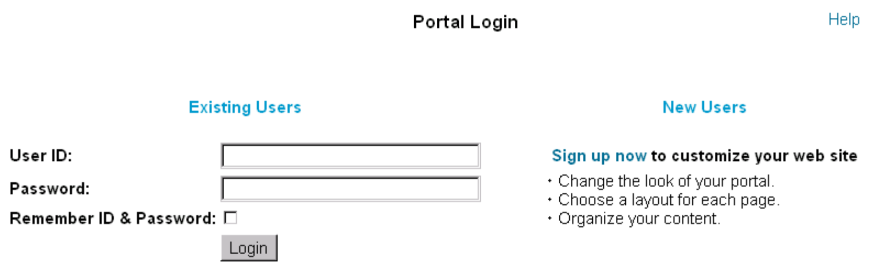
The Stock Portal Web application contains a number of JSPs that handle visitor login and delegate authentication activities to the WebLogic Server. These JSPs are located below the `PORTAL_HOME\applications\portal\stockportal\framework\security` directory.

Logging Into the Portal

The `portal.jsp` template does not automatically prompt a visitor for a username and password. Rather, the `portal.jsp` template requires visitors to initiate the login process by clicking the Login button. When a visitor clicks the Login link on the `portal.jsp` template, the following processing takes place:

- The `login.jsp` template, shown in Figure 3-4, is displayed to the user.

Figure 3-4 Portion of the login.jsp Template


Portal Login

Help

Existing Users

User ID:

Password:

Remember ID & Password: ☐

Login

New Users

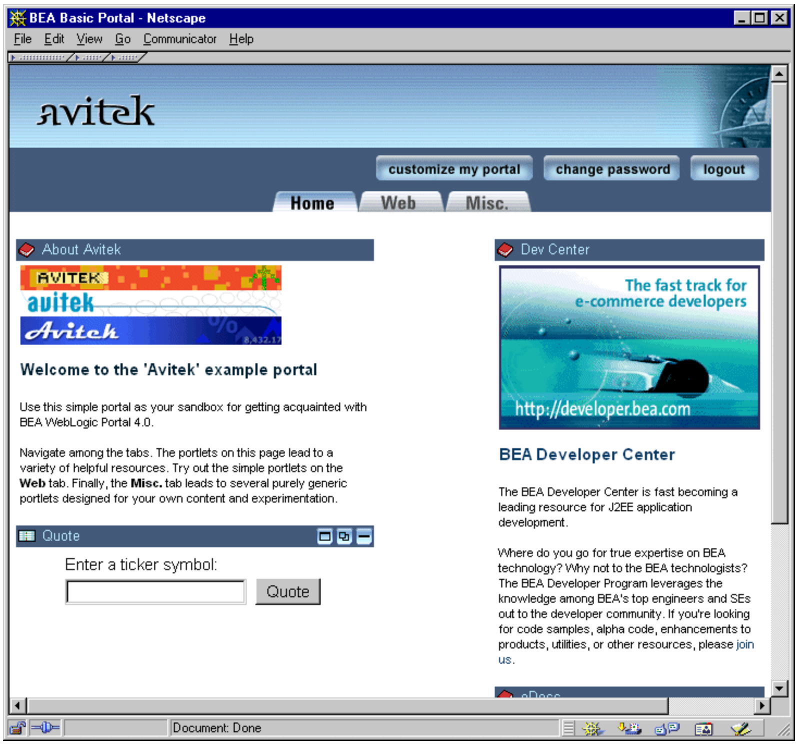
Sign up now to customize your web site

- Change the look of your portal.
- Choose a layout for each page.
- Organize your content.

- When the user clicks the Login button on the `login.jsp` template, security for the Stock Portal sample Web application is turned over to the Webflow mechanism. The Webflow mechanism uses a number of Input Processors (and a few other JSP templates such as `badlogin.jsp`) to complete the login and/or registration process. Details about the Webflow mechanism and its role in portal security are provided in "Webflow and Portal Security" on page 3-20.

Once a visitor has successfully logged into the Stock Portal Web application, that visitor is considered a registered user and presented with additional links (Customize My Portal, Change Password, and Logout) on the `portal.jsp` template, as shown in Figure 3-5. Registered users may now customize the portal by selecting pages, the layouts or portlets that appear within those pages, or by applying portlet skins to change the look of their portal.

Figure 3-5 Avitek Stock Portal Home Page



Welcome to the 'Avitek' example portal

Use this simple portal as your sandbox for getting acquainted with BEA WebLogic Portal 4.0.

Navigate among the tabs. The portlets on this page lead to a variety of helpful resources. Try out the simple portlets on the **Web** tab. Finally, the **Misc.** tab leads to several purely generic portlets designed for your own content and experimentation.

BEA Developer Center

The BEA Developer Center is fast becoming a leading resource for J2EE application development.

Where do you go for true expertise on BEA technology? Why not to the BEA technologists? The BEA Developer Program leverages the knowledge among BEA's top engineers and SEs out to the developer community. If you're looking for code samples, alpha code, enhancements to products, utilities, or other resources, please join us.

Note: The pages, layouts, portlets, and skins available to a registered user are determined by administrators via visitor entitlements. For more information about visitor entitlements, see "Visitor Entitlements" on page 6-9.

Webflow and Portal Security

Out of the box, WebLogic Portal (and thus, the Stock Portal Web application) provides several security-related features, including those that allow a user to login, logout, autologin, change their password, and so on. Each of these security-related features is driven by an underlying Webflow. This Webflow contains a number of Input Processors, each of which are designed to perform discrete, specific tasks related to security. Because of this modular structure, you can easily customize security by adding other Input Processors to the Webflow, or by removing existing Input Processors from the Webflow.

Note: If you are unfamiliar with the Webflow mechanism, see the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* documentation.

Portal security is handled by two separate Webflow files: `security.wf` and `user_account.wf`, which are located in the `PORTAL_HOME\applications\portal-project\application-sync\webapps\stockportal` directory. The `security.wf` file contains the fundamental security logic and should <u>not</u> be modified. The `user_account.wf` file is where you will make any changes to the Webflow that affects portal security.

Note: Changes made to any `.wf` file should be done using the Webflow Editor. For more information, see " Using the Webflow and Pipeline Editors."

Security-Related Use Cases

The following use cases are provided to help you understand how WebLogic Portal handles security via the Webflow mechanism by default. It is recommended that you read each use case in sequence the first time through. This, along with an understanding of the individual Input Processors, should enable you to start modifying the Webflow that affects portal security.

Notes: Details about any Input Processor in this section are described in more detail in the *Javadoc*. All JSPs described in this section can be located in the `PORTAL_HOME\applications\portal\stockportal\framework\security` directory.

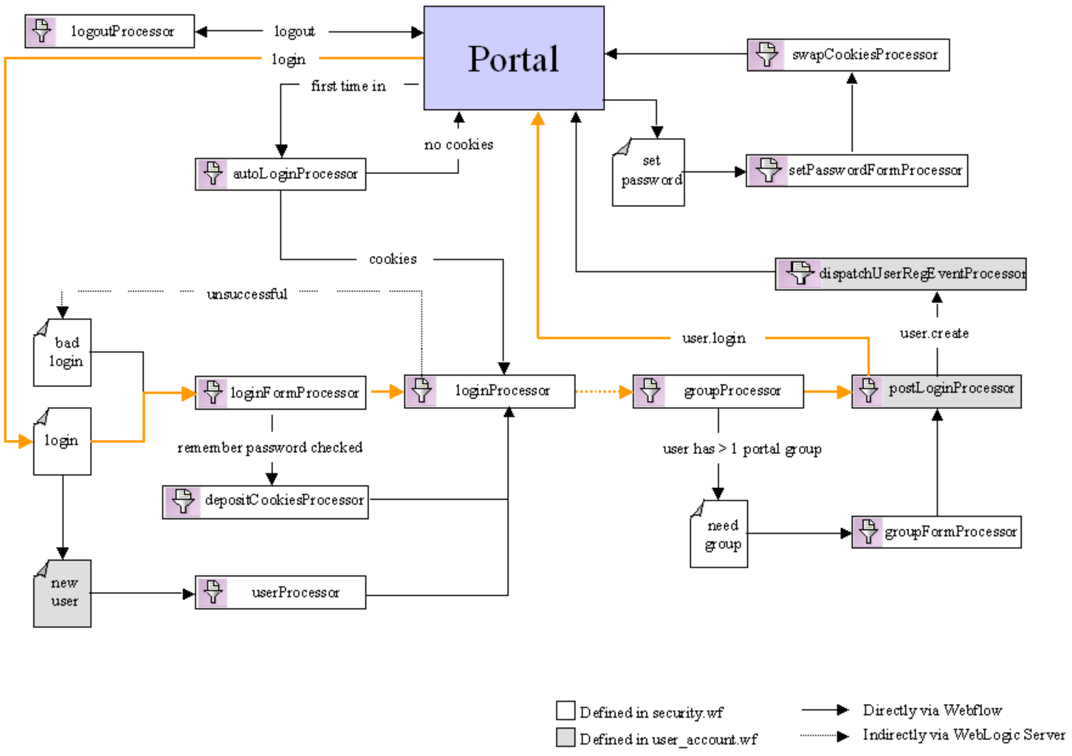
The use cases described in this section are:

- Existing User Logs In (Single Group, Autologin Disabled)
- Existing User Logs In (Multiple Groups, Autologin Disabled)
- Existing User Logs In (Multiple Groups, Autologin Enabled)
- Existing User Changes Password
- Existing User Logs Out
- New User Creates Account

Existing User Logs In (Single Group, Autologin Disabled)

In this simplest of use cases, the user has already registered with the portal Web site and is logging in as part of a return visit. The user belongs to only one portal group, and autologin has not been enabled for the portal Web application. Figure 3-6 illustrates the appropriate path through the Webflow for this use case.

Figure 3-6 Webflow For Existing User Login (Single Group, Autologin Disabled)



In this case, the user starts at `portal.jsp`, where they click the Login link. As shown in Figure 3-6, this action causes the Webflow to display `login.jsp`.

The Webflow forwards information the user entered into the username and password fields on `login.jsp` to the `loginFormProcessor` Input Processor, where it is validated against criteria such as appropriate length, valid characters, and so on. (This criteria is specified in context parameters in the application's `web.xml` deployment descriptor. For more information, see "Contents of the Stock Portal Web Application's Deployment Descriptors" on page 3-32.) If the information entered into the fields is invalid, the `loginFormProcessor` Input Processor indicates that the Webflow should return the user to `login.jsp` to make corrections. If the information is valid, the `loginFormProcessor` Input Processor indicates that the Webflow should forward to the `loginProcessor` Input Processor.

The function of the `loginProcessor` Input Processor is to actually log the user into the system. This activity requires a call to the underlying WebLogic Server with a `j_security_check`. When WebLogic Server is finished, control is sent back to the portal Webflow.

If the username/password combination itself is invalid and the user cannot be logged in, the `loginProcessor` Input Processor indicates that the Webflow should direct the user to `badlogin.jsp` to correct the information. If the `loginProcessor` Input Processor finds no problems with the user's information, the `loginProcessor` Input Processor indicates that the Webflow should forward to the `groupProcessor` Input Processor.

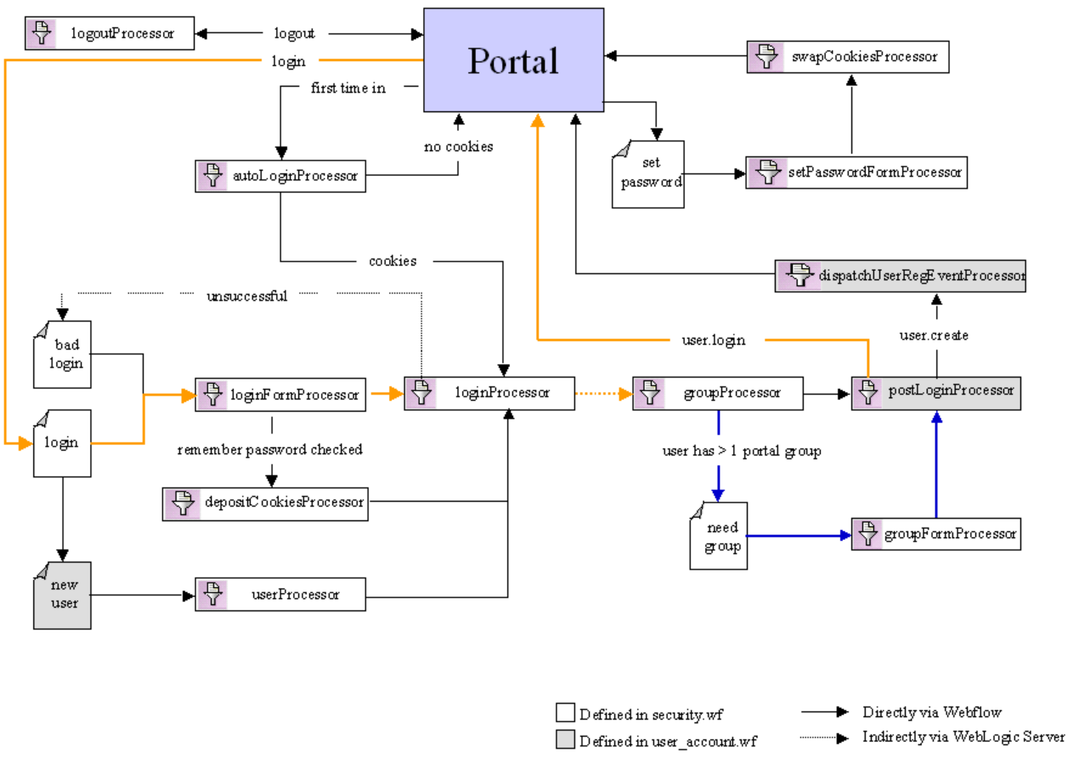
In this use case, the `groupProcessor` Input Processor determines that the user has only one portal group, and simply indicates to the Webflow that control should be forwarded to the `postLoginProcessor` Input Processor.

The `postLoginProcessor` Input Processor is the last mandatory Input Processor in the Webflow. Arriving at the `postLoginProcessor` Input Processor means that the user is guaranteed to be logged in and that a portal group is assigned to the user. The `postLoginProcessor` Input Processor performs the final set up work for the user by placing the correct profile in the HTTP session, then returns `user.login` to the Webflow, indicating that `portal.jsp` should be redisplayed. When `portal.jsp` is redisplayed to the user, the user is considered logged in and will see any predefined customizations.

Existing User Logs In (Multiple Groups, Autologin Disabled)

This use case is, for the most part, the same as that shown in "Existing User Logs In (Single Group, Autologin Disabled)" on page 3-21, with one exception. Because the user belongs to multiple portal groups, the `groupProcessor` Input Processor must do more than just indicate to the Webflow that control should be forwarded to the `postLoginProcessor` Input Processor. This alternative flow (highlighted in blue) is shown in Figure 3-7.

Figure 3-7 Webflow for Existing Login (Multiple Groups, Autologin Disabled)



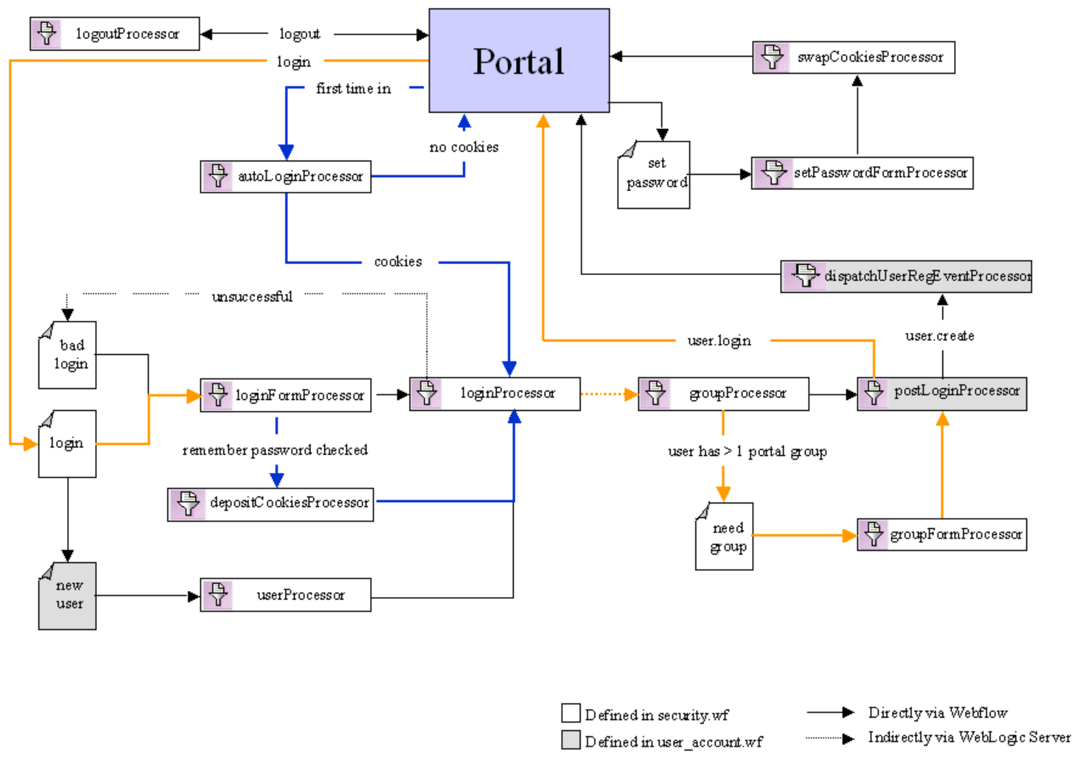
In this use case, the `groupProcessor` Input Processor determines that the user belongs to multiple portal groups, and indicates to the Webflow that `need_group.jsp` should be displayed. The `need_group.jsp` allows the user to select the group with which their login should be associated.

When the user moves from `need_group.jsp`, Webflow forwards to the `groupFormProcessor` Input Processor to validate their selection. If no selection is made, the `groupFormProcessor` Input Processor indicates that the user should be sent back to `need_group.jsp`. If the user has made a selection, the `groupFormProcessor` Input Processor indicates that control should be passed to the `postLoginProcessor` Input Processor with the group selected, and flow continues as described in “Existing User Logs In (Single Group, Autologin Disabled)” on page 3-21.

Existing User Logs In (Multiple Groups, Autologin Enabled)

This use case is a modification of the “Existing User Logs In (Multiple Groups, Autologin Disabled)” previously described. Because autologin is enabled in this use case, the first part of the Webflow is different. This alternative flow (highlighted in blue) is shown in Figure 3-8.

Figure 3-8 Webflow for User Login (Multiple Groups, Autologin Enabled)



There are two prerequisites for the autologin feature:

- Autologin must be enabled by setting the `PORTAL_AUTO_LOGIN` context parameter in your portal Web application’s `web.xml` deployment descriptor to `true`. Recall that the `web.xml` file is located in the Web application’s `Web-inf` directory. For more information, see “Contents of the Stock Portal Web Application’s Deployment Descriptors” on page 3-32.

- Autologin can be used only when the autologin option is selected by a user on `login.jsp` (by clicking the Remember ID & Password check box).

Thus, when a user first comes to the portal Web site, `portal.jsp` relies upon an internal preprocessor (not shown in the figure) to determine whether the user's session is new *and* whether autologin for the portal Web application has been enabled in the `web.xml` deployment descriptor. If the session is new and autologin is enabled, `portal.jsp` indicates to the Webflow that control should be fowarded to the `autoLoginProcessor` Input Processor, which attempts to locate autologin cookies on the user's machine.

If cookies are found, this means that the user has previously selected the autologin option on `login.jsp`. The `autoLoginProcessor` Input Processor then indicates to the Webflow that control should be fowarded to the `loginProcessor` Input Processor, and the flow continues as described in "Existing User Logs In (Multiple Groups, Autologin Disabled)" on page 3-23. If no cookies are found, this means that the user has not selected the autologin option on `login.jsp`, and the Webflow returns the user to `portal.jsp`.

If the user is sent back to `portal.jsp`, user login takes place the same way as that described in "Existing User Logs In (Multiple Groups, Autologin Disabled)" on page 3-23. (The internal preprocessor in `portal.jsp` is aware that the user cannot use autologin and therefore does not try this path again.) However, if the user clicks the Remember ID & Password check box on `login.jsp` during this process, the `loginFormProcessor` Input Processor detects this and returns a value to the Webflow indicating that control should be forwarded to the `depositCookiesProcessor` Input Processor. Then, the `depositCookiesProcessor` Input Processor deposits two cookies on the user's machine: their encrypted username, and their encrypted password.

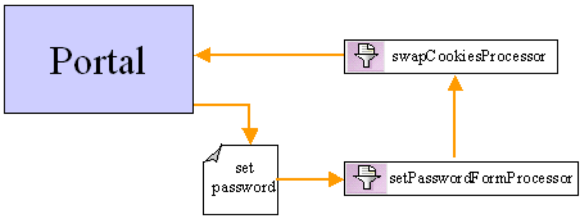
Once the cookies are placed, the `depositCookiesProcessor` Input Processor indicates to the Webflow that control should be forwarded to the `loginProcessor` Input Processor, and the flow continues as usual.

Caution: If the user explicitly logs out of the system by clicking the Logout link (see "Existing User Logs Out" on page 3-27), autologin does <u>not</u> result in automatic re-login the next time the user visits the portal Web site. Rather, autologin causes the user to be automatically re-logged in *if their session times out*.

Existing User Changes Password

The portion of the Webflow for changing user passwords (as defined in the `security.wf` file) is shown in Figure 3-9.

Figure 3-9 Webflow for Changing User Passwords



When a user clicks the Change Password link on `portal.jsp`, the Webflow displays the `set_password.jsp`. When the user moves from `set_password.jsp`, the Webflow fowards to the `setPasswordFormProcessor` Input Processor to validate the information entered.

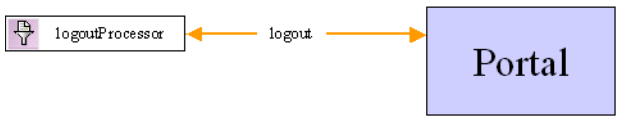
If the information entered is not valid (in length, allowed characters, and so on), the `setPasswordFormProcessor` Input Processor indicates to the Webflow that the user should be sent back to `set_password.jsp`. (The criteria for validation is specified in context parameters in the application's `web.xml` deployment descriptor. For more information, see "Contents of the Stock Portal Web Application's Deployment Descriptors" on page 3-32.) If the user has entered valid information, the `setPasswordFormProcessor` Input Processor makes the changes to the password in the system, then indicates to the Webflow that control should be forwarded to the `swapCookiesProcessor` Input Processor.

If autologin cookies have been set for the user's username and encrypted password on their machine, the `swapCookiesProcessor` Input Processor updates the cookies with the new username and encrypted password information, and indicates to the Webflow that `portal.jsp` should be displayed. If no autologin cookies exist, the `swapCookiesProcessor` Input Processor simply indicates to the Webflow that `portal.jsp` should be displayed.

Existing User Logs Out

The Webflow for logging out existing users (as defined in the `security.wf` file) is shown in Figure 3-10.

Figure 3-10 Webflow for Logging Out Existing Users



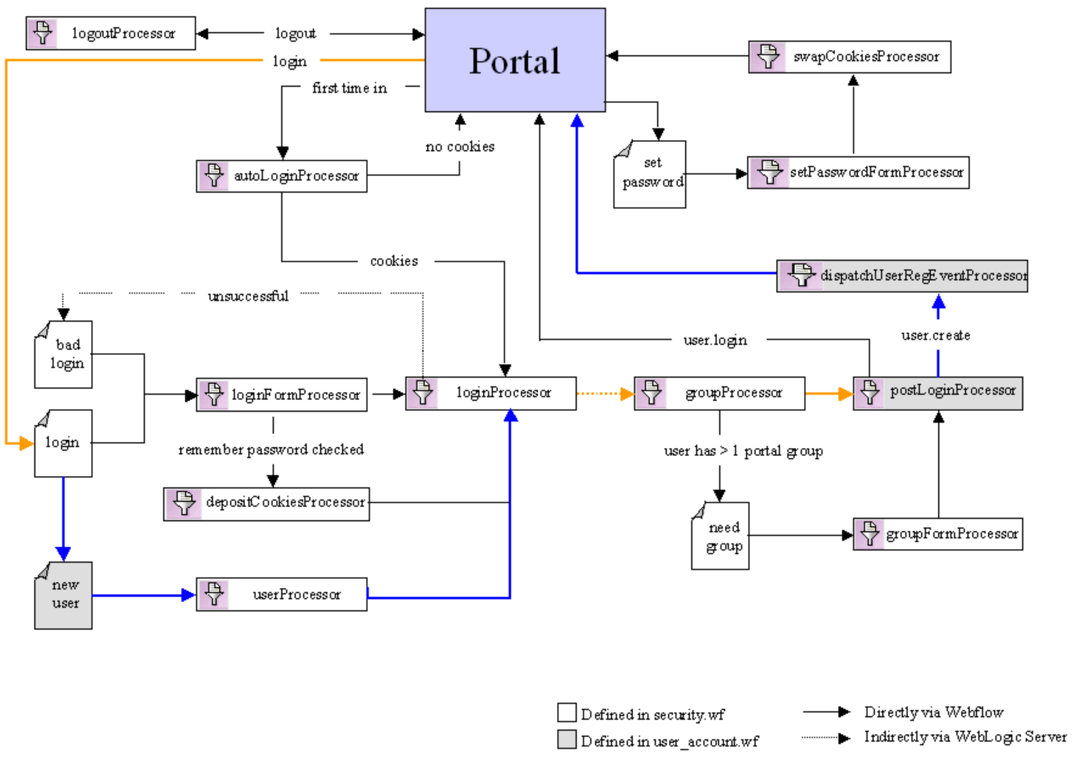
When a user explicitly clicks the Logout link on `portal.jsp`, the Webflow passes control to the `logoutProcessor` Input Processor. The `logoutProcessor` Input Processor invalidates the user's session, clears the autologin cookies (if they exist), and indicates to the Webflow that the default version of `portal.jsp` (that is, the non-personalized version) should be displayed to the user.

New User Creates Account

In this final use case, the user is brand new to the portal Web site, and has never logged in before. However, the user would like to be able to personalize `portal.jsp`, so they click the Login link. This causes `login.jsp` to be displayed as in previous use cases.

However, once the user arrives at `login.jsp`, the flow of control is geared toward new user registration, as shown (highlighted in blue) in Figure 3-11.

Figure 3-11 Webflow for Creating New User Accounts



From `login.jsp`, the user indicates they are a new user and would like to register by clicking Sign Up Now, instead of entering a pre-existing username and password combination. Webflow then passes control to `new_user.jsp`, which gathers information about the user (such as their username and password). When the user has completed the form fields on `new_user.jsp`, the Webflow invokes the `userProcessor` Input Processor.

The `userProcessor` Input Processor performs two functions: it validates the username and password form fields against criteria such as length, valid characters, and so on, and it actually creates the new user in the system (security realm). (The criteria for validation is specified in context parameters in the application's `web.xml` deployment descriptor. For more information, see "Contents of the Stock Portal Web Application's Deployment Descriptors" on page 3-32.)

When the `userProcessor` Input Processor has completed its tasks, it indicates to Webflow that control should be passed to the `loginProcessor` Input Processor. Once the `loginProcessor` Input Processor is invoked, the flow proceeds as described in previous use cases, up until the `postLoginProcessor` Input Processor completes execution. In the case of new user registration, the `postLoginProcessor` Input Processor returns a value of `user.create` to the Webflow, which results in a call to an additional Input Processor named `dispatchUserRegEventProcessor` (instead of returning `user.login` to the Webflow and redisplaying `portal.jsp` as described in previous use cases). The `dispatchUserRegEventProcessor` Input Processor dispatches an event to indicate that a new user has registered with the portal Web site, then indicates to the Webflow that `portal.jsp` should be redisplayed.

Note: For more information about events and behavior tracking, see the *Guide to Events and Behavior Tracking* documentation.

Modifying the Portal Security Webflow: An Example

Figure 3-13 and Figure 3-13 illustrate one way the portal security Webflow may be modified (relevant portions only). In this example, the Webflow is extended to allow users to enter, view and update additional account information (that is, besides their username and password handled by `userProcessor` Input Processor described in previous use cases).

Note: The JSPs and Input Processors described in this section are just examples, and are not included with the WebLogic Portal product. They are extensions to the portal security Webflow and as such, are to be coded by your organization's HTML/JSP and Java/EJB developers according to your requirements.

Figure 3-12 Example of Extending the Security Portal Webflow: Part 1

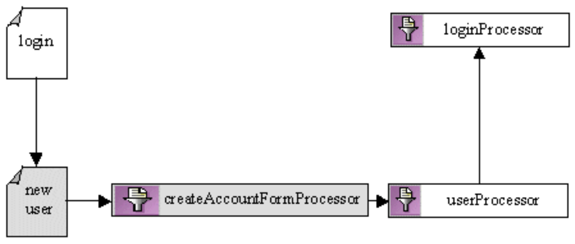
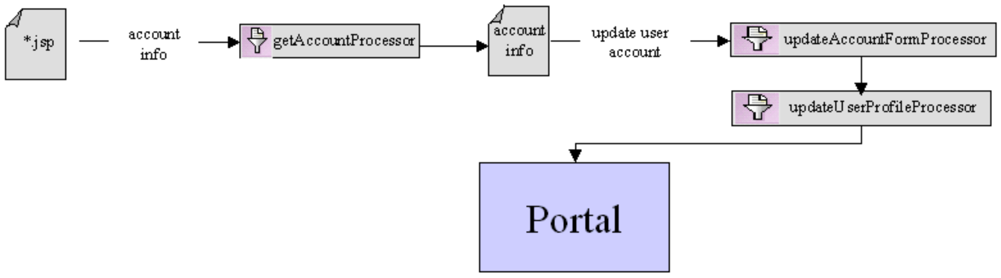


Figure 3-12 shows the addition of the `createAccountFormProcessor` Input Processor as part of the new user registration process. This Input Processor may allow users to enter information other than the username and password information (for which the `userProcessor` Input Processor is responsible). It may also perform validation on the additional form fields.

Figure 3-13 Example of Extending the Security Portal Webflow: Part 2



As shown in Figure 3-13, a related modification to the Webflow might allow the user to click an Account Info link from any JSP, which causes the Webflow to pass control to the `getAccountProcessor` Input Processor. The `getAccountProcessor` Input Processor obtains the user's profile information, bundles it into a class, and places it into the Pipeline Session as a Pipeline Session-scoped property. The `getAccountProcessor` Input Processor then indicates to the Webflow that control should be forwarded to an account information JSP, which displays the information to the user.

Note: For more information about the Pipeline Session and property scoping, see "Pipeline Session Internals" in the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* documentation.

If the user then clicks an Update Account link (after making changes to their information), the Webflow calls the `updateAccountFormProcessor` Input Processor to validate the form fields. If there are problems with the form validation, the Webflow will send the user back to the Account Info page to rectify the problem(s). If there are no problems with the form validation, the Webflow passes control to the `updateUserProfileProcessor` Input Processor, which actually updates the user's information in the database.

Contents of the Stock Portal Web Application's Deployment Descriptors

This section provides detailed information about the security-related XML elements in the Stock Portal Web application's deployment descriptor files. It is intended as a reference. For background conceptual information about the security elements listed in this section, see "The web.xml Deployment Descriptors" on page 2-12 and "The weblogic.xml Deployment Descriptors" on page 2-13.

web.xml

Along with other information, the `web.xml` deployment descriptor for the Stock Portal Web application contains several sets of XML elements for implementing the J2EE declarative security model. The contents of these elements are shown in the following code listings:

- Declarations of Secure Links
- Session Configuration
- Declaration of Secure JSPs
- Login Configuration
- Autologin and Form Field Validation Settings

Listing 3-9 Declarations of Secure Links

```
<context-param>
   <param-name>HTTPS_URL_PATTERNS</param-name>
   <param-value>/security/login.jsp, /security/new_user.jsp,
    /security/userProcessor.inputprocessor,
    /security/loginProcessor.inputprocessor</param-value>
</context-param>
```

Listing 3-10 Session Configuration

```
<session-config>
   <session-timeout>15</session-timeout>
</session-config>
```

Listing 3-11 Declaration of Secure JSPs

```
<security-constraint>
   <web-resource-collection>

       <web-resource-name>Pages which require login
       </web-resource-name>
       <description>Pages which require login</description>
       <url-pattern>/security/need_group.jsp</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
   </web-resource-collection>

</security-constraint>
```

Listing 3-12 Login Configuration

```
<login-config>
   <auth-method>FORM</auth-method>
   <realm-name>default</realm-name>

   <form-login-config>
       <form-login-page>/framework/security/login.jsp
       </form-login-page>
       <form-error-page>/framework/security/badlogin.jsp
       </form-error-page>
   </form-login-config>

</login-config>
```

Listing 3-13 Autologin and Form Field Validation Settings

```
<!--
 If set to true Portal will perform auto login via cookies
-->

<context-param>
   <param-name>PORTAL_AUTO_LOGIN</param-name>
   <param-value>true</param-value>
</context-param>

<!--
 Override the default minimum password length
-->

<!--
<context-param>
   <param-name>PORTAL_MIN_PASSWORD_LEN</param-name>
   <param-value>4</param-value>
</context-param>
-->

<!--
 Override the default maximum password length
-->

<!--
<context-param>
   <param-name>PORTAL_MAX_PASSWORD_LEN</param-name>
   <param-value>10</param-value>
</context-param>
-->

<!--
 Override the default minimum username length
-->

<!--
<context-param>
   <param-name>PORTAL_MIN_USERNAME_LEN</param-name>
   <param-value>4</param-value>
</context-param>
-->

<!--
 Override the default maximum username length

<context-param>
   <param-name>PORTAL_MAX_USERNAME_LEN</param-name>
   <param-value>10</param-value>
```

```
</context-param>
-->

<!--
 Override the default timeout for autologin cookies (in seconds)

<context-param>
   <param-name>PORTAL_AUTOLOGIN_COOKIE_TIMEOUT</param-name>
   <param-value>172800</param-value>
</context-param>
-->
```

weblogic.xml

Along with other information, the `weblogic.xml` deployment descriptor for the Stock Portal Web application contains several sets of XML elements for implementing the J2EE declarative security model. The contents of these elements are shown in the following code listings:

- Security-Role Mappings
- Check Authentication on JSP Forwarding
- Cookies

Listing 3-14 Security-Role Mappings

```
<security-role-assignment>
   <role-name>SystemAdminRole</role-name>
   <principal-name>SystemAdministrator</principal-name>
   <principal-name>system</principal-name>
</security-role-assignment>
```

Listing 3-15 Check Authentication on JSP Forwarding

```
<container-descriptor>
   <check-auth-on-forward />
</container-descriptor>
```

Listing 3-16 Cookies

```
<session-descriptor>
   <session-param>
       <param-name>CookieName</param-name>
       <param-value>JSESSIONID_STOCKPORTAL</param-value>
   </session-param>

   <session-param>
       <param-name>CookiePath</param-name>
       <param-value>/stockportal</param-value>
   </session-param>

   <session-param>
       <param-name>TimeoutSecs</param-name>
       <param-value>300</param-value>
   </session-param>

</session-descriptor>
```

4 Security in the WebLogic Portal Enterprise JavaBeans

As described in Chapter 2, "Security Roles and Deployment Descriptors," each Enterprise JavaBean (EJB) Java ARchive (JAR) file has two associated deployment descriptors: `ejb-jar.xml` and `weblogic-ejb-jar.xml`. These XML files contain elements that register an application's individual JavaBeans with appropriate security constraints. Since you will use the services provided by these EJBs a starting point for developing your own applications, this topic describes the contents of the deployment descriptors for each JAR in WebLogic Portal. Therefore, this topic includes the following sections:

- Note About the weblogic-ejb-jar.xml Deployment Descriptors in WebLogic Portal
- campaign.jar
- campaignws.jar
- catalogws.jar
- customer.jar
- document.jar
- ebusiness.jar
- ejbadvisor.jar
- events.jar

- ldapprofile.jar
- mail.jar
- payment.jar
- petflow.jar
- petStore_EJB.jar
- pipeline.jar
- placeholder.jar
- portal.jar
- property.jar
- propertysetws.jar
- rules.jar
- stockportal.jar
- tax.jar
- usermgmt.jar
- wlcsSample.jar

Note About the weblogic-ejb-jar.xml Deployment Descriptors in WebLogic Portal

Because the security elements in the `weblogic-ejb-jar.xml` deployment descriptors for each WebLogic Portal EJB JAR may be the same, a listing of this file's contents for each EJB JAR file will not be shown. For all EJB JARs discussed in this topic, the `weblogic-ejb-jar.xml` deployment descriptor may contain one or more of the following security-role assignments:

```
<security-role-assignment>
   <role-name>CustomerRole</role-name>
   <principal-name>wlcs_customer</principal-name>
</security-role-assignment>

<security-role-assignment>
   <role-name>DelegatedAdminRole</role-name>
   <principal-name>DelegatedAdministrator</principal-name>
</security-role-assignment>

<security-role-assignment>
   <role-name>EveryoneRole</role-name>
   <principal-name>everyone</principal-name>
</security-role-assignment>

<security-role-assignment>
   <role-name>SystemAdminRole</role-name>
   <principal-name>SystemAdministrator</principal-name>
   <principal-name>system</principal-name>
</security-role-assignment>
```

campaign.jar

The `campaign.jar` file is a collection of EJBs that provide the portal campaign and user interaction services. The `campaign.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- ActionService
- CampaignService
- ScenarioService

Table 4-1 Security Settings for the ActionService EJB

Name	ActionService
Description	Provides the executable actions for the Campaign services (that is, the mail action, the ads action, and the offer discount action).

Table 4-1 Security Settings for the ActionService EJB (Continued)

Class	`com.bea.campaign.action.internal.ActionServiceBean`
Home Interface	`com.bea.campaign.action.ActionServiceHome`
Remote Interface	`com.bea.campaign.action.ActionService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-2 Security Settings for the CampaignService EJB

Name	CampaignService
Description	The main entry point to the Campaign service.
Class	`com.bea.campaign.internal.CampaignServiceImpl`
Home Interface	`com.bea.campaign.CampaignServiceHome`
Remote Interface	`com.bea.campaign.CampaignService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `handleEvent`, `isActive`, `getScenarioService`, `getAdService`, `getAdBucketService`, `getMailService`. The `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-3 Security Settings for the ScenarioService EJB

Name	ScenarioService
Description	Executes campaign scenarios.
Class	`com.bea.campaign.internal.ScenarioServiceImpl`
Home Interface	`com.bea.campaign.ScenarioServiceHome`
Remote Interface	`com.bea.campaign.ScenarioService`
Type	Stateless session bean
Security Roles and Role Assignments	The `AnonymousRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `locateService`, `handleEvent`, `getRulesManager`, `getUserEndStates`, `setUserEndState`. The `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

campaignws.jar

The `campaignws.jar` file contains an EJB that supports a Web service that the E-Business Control Center uses to populate ad query and e-mail picklists. The `campaignws.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- CampaignWebService

Table 4-4 Security Settings for the CampaignWebService EJB

Name	CampaignWebService
Description	Supports a Web service that the E-Business Control Center uses to populate ad query and e-mail picklists.

Table 4-4 Security Settings for the CampaignWebService EJB (Continued)

Class	`com.bea.campaign.webservice.internal.CampaignWebServiceImpl`
Home Interface	`com.bea.campaign.webservice.CampaignWebServiceHome`
Remote Interface	`com.bea.campaign.webservice.CampaignWebService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

catalogws.jar

The `catalogws.jar` file contains an EJB that provides a Web service that the E-Business Control Center uses to browse the product catalog. The `catalogws.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- CatalogWebService

Table 4-5 Security Settings for the CatalogWebService EJB

Name	CatalogWebService
Description	Supports a Web service that the E-Business Control Center uses to browse the product catalog.
Class	`com.beasys.commerce.ebusiness.catalog.webservice.CatalogWSImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.webservice.CatalogWSHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.webservice.CatalogWS`
Type	Stateless session bean

Table 4-5 Security Settings for the CatalogWebService EJB (Continued)

Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`

customer.jar

The `customer.jar` file is a collection of EJBs that supports the `com.beasys.commerce.ebusiness.customer.Customer` entity. The `customer.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- Customer
- CustomerProfileManager
- CustomerPropertyManager

Table 4-6 Security Settings for the Customer EJB

Name	Customer
Description	Encapsulates customer profile data for registered customers of the Commerce services.
Class	`com.beasys.commerce.ebusiness.customer.CustomerImpl`
Home Interface	`com.beasys.commerce.ebusiness.customer.CustomerHome`
Remote Interface	`com.beasys.commerce.ebusiness.customer.Customer`
Type	Entity bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Remote interfaces.
Security-Role References	`AnonymousRole`

Table 4-7 Security Settings for the CustomerProfileManager EJB

Name	CustomerProfileManager
Description	Extends the ProfileManager to access the Customer profile information. See the *Javadoc* for a complete description of the ProfileManager class.
Class	`com.bea.commerce.ebusiness.customer.internal.` `CustomerProfileManagerImpl`
Home Interface	`com.bea.commerce.ebusiness.customer.` `CustomerProfileManagerHome`
Remote Interface	`com.bea.commerce.ebusiness.customer.CustomerProfileManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	`DelegatedAdminRole`, `SystemAdminRole`

Table 4-8 Security Settings for the CustomerPropertyManager EJB

Name	CustomerPropertyManager
Description	Manages properties associated with Customer and delegates calls to the CustomerEJB. This class handles property sets for "CustomerProperties", and its mappings can be found in `usermgmt-ejb-jar.xml` deployment descriptor.
Class	`com.bea.commerce.ebusiness.customer.internal.` `CustomerPropertyManagerBean`
Home Interface	`com.bea.commerce.ebusiness.customer.` `CustomerPropertyManagerHome`
Remote Interface	`com.bea.commerce.ebusiness.customer.CustomerPropertyManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Remote interfaces.

Table 4-8 Security Settings for the CustomerPropertyManager EJB (Continued)

Security-Role References	None

document.jar

The `document.jar` file contains an EJB that supports document management services (searching, retrieval, and schemas). The `document.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- DocumentManager

Table 4-9 Security Settings for the DocumentManager EJB

Name	DocumentManager
Description	Supports document management services.
Class	`com.bea.p13n.content.document.internal.SPIFastDocumentManagerImpl`
Home Interface	`com.bea.p13n.content.document.DocumentManagerHome`
Remote Interface	`com.bea.p13n.content.document.DocumentManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

ebusiness.jar

The `ebusiness.jar` file is a collection of EJBs that support Commerce services, including product catalog, order, tax calculation, shipping, payment, and supporting EJB Pipeline Components. The `ebusiness.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- CatalogManager
- CatalogQueryManager
- CategoryManager
- CustomDataManager
- Decryptor
- Discount
- DiscountAssociation
- DiscountAssociationMgr
- DiscountManagement
- Encryptor
- EntityPropertyManager
- EpmCustomDataManager
- JdbcCatalogQueryManager
- JdbcCategoryManager
- JdbcProductItemManager
- Order
- OrderManager
- PaymentTransaction
- PriceService

- ProductItemManager
- ShippingHelper
- ShippingMethod

Table 4-10 Security Settings for the CatalogManager EJB

Name	CatalogManager
Description	The central entry-point into the catalog management system—from it you can retrieve the other stateless session beans that compose the sub-systems of the catalog management system.
Class	`com.beasys.commerce.ebusiness.catalog.CatalogManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.CatalogManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.CatalogManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `CustomerRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `getCategoryManager`, `getProductItemManager`, `getCatalogQueryManager`, `getCustomDataManager`, `createCatalogRequest`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AdministrativeRole`, `AnonymousRole`, `CustomerRole`

Table 4-11 Security Settings for the CatalogQueryManager EJB

Name	CatalogQueryManager
Description	Responsible for executing simple keyword-based and complex expression-based queries against the catalog and returning ProductItems.
Class	`com.beasys.commerce.ebusiness.catalog.service.query.` `CatalogQueryManagerImpl`

Table 4-11 Security Settings for the CatalogQueryManager EJB (Continued)

Home Interface	`com.beasys.commerce.ebusiness.catalog.service.query.CatalogQueryManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.query.CatalogQueryManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-12 Security Settings for the CategoryManager EJB

Name	CategoryManager
Description	Responsible for managing the assignment and caching of ProductItems into Categories to impose a hierarchical structure on the catalog.
Class	`com.beasys.commerce.ebusiness.catalog.service.category.CategoryManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.category.CategoryManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.category.CategoryManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `CustomerRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `getCatalogManager`, `getItems`, `getItemKeys`, `getParent`, `getAncestors`, `getRootCategory`, `getCategoryCount`, `getSubCategories`, `getSubCategoryKeys`, `getSiblings`, `getSiblingKeys`, `getCategories`, `getCategoryKeys`, `getItemCount`, `getSubCategoryCount`, `getSiblingCount`, `getCategory`, `getCategories`, `getItemCategories`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.

Table 4-12 Security Settings for the CategoryManager EJB (Continued)

Security-Role References	`AdministrativeRole`, `AnonymousRole`, `CustomerRole`

Table 4-13 Security Settings for the CustomDataManager EJB

Name	CustomDataManager
Description	Responsible for storing the custom attributes assigned to ProductItems in the catalog.
Class	`com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-14 Security Settings for the Decryptor EJB

Name	Decryptor
Description	Decrypts a customer's encrypted credit card information when this information needs to be sent to the payment system.
Class	`com.beasys.commerce.ebusiness.security.DecryptorImpl`
Home Interface	`com.beasys.commerce.ebusiness.security.DecryptorHome`
Remote Interface	`com.beasys.commerce.ebusiness.security.Decryptor`
Type	Stateless session bean

Table 4-14 Security Settings for the Decryptor EJB (Continued)

Security Roles and Role Assignments	`CustomerRole` and `AdministrativeRole` can access all methods in the bean’s Home and Remote interfaces.
Security-Role References	`CustomerRole`, `AdministrativeRole`

Table 4-15 Security Settings for the Discount EJB

Name	Discount
Description	Provides access to discount definitions stored in the `DISCOUNT` database table.
Class	`com.bea.commerce.ebusiness.discount.mgmt.internal.DiscountImpl`
Home Interface	`com.bea.commerce.ebusiness.discount.mgmt.internal.DiscountHome`
Remote Interface	`com.bea.commerce.ebusiness.discount.mgmt.internal.Discount`
Type	Entity bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole` and `AdministrativeRole` can access all methods in the bean’s Home and Remote interfaces.
Security-Role References	None

Table 4-16 Security Settings for the DiscountAssociation EJB

Name	DiscountAssociation
Description	Provides access to data in the `DISCOUNT_ASSOCIATION` database table.
Class	`com.bea.commerce.ebusiness.discount.association.internal.DiscountAssocImpl`
Home Interface	`com.bea.commerce.ebusiness.discount.association.internal.DiscountAssocHome`

Table 4-16 Security Settings for the DiscountAssociation EJB (Continued)

Remote Interface	`com.bea.commerce.ebusiness.discount.association.internal.DiscountAssoc`
Type	Entity bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole` and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-17 Security Settings for the DiscountAssociationMgr EJB

Name	DiscountAssociationMgr
Description	Provides methods for creating, deleting, querying, and updating discount associations.
Class	`com.bea.commerce.ebusiness.discount.association.internal.DiscountAssociationMgrImpl`
Home Interface	`com.bea.commerce.ebusiness.discount.association.DiscountAssociationMgrHome`
Remote Interface	`com.bea.commerce.ebusiness.discount.association.DiscountAssociationMgr`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface, and the following methods on the bean's Remote interface: `updateUsesGlobal`, `updateUses`, `queryGlobal`, `queryUser`, `add`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`, `AdministrativeRole`

Table 4-18 Security Settings for the DiscountManagement EJB

Name	DiscountManagement
Description	Provides methods for obtaining discounnt definitions.
Class	`com.bea.commerce.ebusiness.discount.mgmt.internal.DiscountMgmtImpl`
Home Interface	`com.bea.commerce.ebusiness.discount.mgmt.DiscountMgmtHome`
Remote Interface	`com.bea.commerce.ebusiness.discount.mgmt.DiscountMgmt`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface, and the following methods on the bean's Remote interface: `getGlobalDiscounts`, `getDiscountById`, `getDiscountsById`, `getDiscountByName`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`, `AdministrativeRole`

Table 4-19 Security Settings for the Encryptor EJB

Name	Encryptor
Description	Encrypts a customer's credit card information.
Class	`com.beasys.commerce.ebusiness.security.EncryptorImpl`
Home Interface	`com.beasys.commerce.ebusiness.security.EncryptorHome`
Remote Interface	`com.beasys.commerce.ebusiness.security.Encryptor`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.

Table 4-19 Security Settings for the Encryptor EJB (Continued)

Security-Role References	`AnonymousRole`

Table 4-20 Security Settings for the EntityPropertyManager EJB

Name	EntityPropertyManager
Description	Responsible for handling the persistence of profile properties in the RDBMS tables.
Class	`com.bea.p13n.property.internal.EntityPropertyManagerImpl`
Home Interface	`com.bea.p13n.property.EntityPropertyManagerHome`
Remote Interface	`com.bea.p13n.property.EntityPropertyManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-21 Security Settings for the EpmCustomDataManager EJB

Name	EpmCustomDataManager
Description	An instance of the EntityPropertyManager EJB, used by the catalog system to store the custom attributes for ProductItems and Categories. The CatalogCustomDataManager delegages to the EpmCustomDataManager.
Class	`com.beasys.commerce.ebusiness.catalog.service.data.EpmCustomDataManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManager`

Table 4-21 Security Settings for the EpmCustomDataManager EJB (Continued)

Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-22 Security Settings for the JdbcCatalogQueryManager EJB

Name	JdbcCatalogQueryManager
Description	A tier-2 stateless session bean that performs queries against ProductItems persisted using JDBC. For more information about the catalog architecture, see *Guide to Building a Product Catalog*.
Class	`com.beasys.commerce.ebusiness.catalog.service.query.` `JdbcCatalogQueryManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.query.` `CatalogQueryManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.query.` `CatalogQueryManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-23 Security Settings for the JdbcCategoryManager EJB

Name	JdbcCategoryManager
Description	A tier-2 stateless session bean that manages the hierarchy of the catalog using JDBC persistence. For more information about the catalog architecture, see *Guide to Building a Product Catalog*.

Table 4-23 Security Settings for the JdbcCategoryManager EJB (Continued)

Class	`com.beasys.commerce.ebusiness.catalog.service.category.JdbcCategoryManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.category.CategoryManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.category.CategoryManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `CustomerRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `getCatalogManager`, `getItems`, `getItemKeys`, `getParent`, `getAncestors`, `getRootCategory`, `getCategoryCount`, `getSubCategories`, `getSubCategoryKeys`, `getSiblings`, `getSiblingKeys`, `getCategories`, `getCategoryKeys`, `getItemCount`, `getSubCategoryCount`, `getSiblingCount`, `getCategory`, `getCategories`, `getItemCategories`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AdministrativeRole`, `AnonymousRole`, `CustomerRole`

Table 4-24 Security Settings for the JdbcProductItemManager EJB

Name	JdbcProductItemManager
Description	A tier-2 stateless session bean that stores ProductItems using JDBC persistence. For more information about the catalog architecture, see *Guide to Building a Product Catalog*.
Class	`com.beasys.commerce.ebusiness.catalog.service.item.JdbcProductItemManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManager`
Type	Stateless session bean

Table 4-24 Security Settings for the JdbcProductItemManager EJB (Continued)

Security Roles and Role Assignments	`AnonymousRole` and `CustomerRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `getCatalogManager`, `getItemCount`, `getKeywords`, `getItemKeys`, `getItems`, `getItem`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AdministrativeRole`, `AnonymousRole`, `CustomerRole`

Table 4-25 Security Settings for the Order EJB

Name	Order
Description	An Order entity represents an order. An order includes one or more order lines, order status, shipping address, shipping cost, price, special instructions, splitting preferences, date of order, payment information, and the customer who placed this order.
Class	`com.beasys.commerce.ebusiness.order.OrderImpl`
Home Interface	`com.beasys.commerce.ebusiness.order.OrderHome`
Remote Interface	`com.beasys.commerce.ebusiness.order.Order`
Type	Entity bean
Security Roles and Role Assignments	`CustomerRole` and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`CustomerRole, AdministrativeRole`

Table 4-26 Security Settings for the OrderManager EJB

Name	OrderManager
Description	Creates an Order and retrieves the OrderValues using a variety of criteria, customerPK, a date range, a SKU, and a Status.

Table 4-26 Security Settings for the OrderManager EJB (Continued)

Class	`com.beasys.commerce.ebusiness.order.OrderManagerBean`
Home Interface	`com.beasys.commerce.ebusiness.order.OrderManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.order.OrderManager`
Type	Stateless session bean
Security Roles and Role Assignments	`CustomerRole` and `AdministrativeRole` can access all methods in the bean's Home interface.
Security-Role References	`CustomerRole, AdministrativeRole`

Table 4-27 Security Settings for the PaymentTransaction EJB

Name	PaymentTransaction
Description	Maintains the state of an individual credit card transaction.
Class	`com.beasys.commerce.ebusiness.payment.PaymentTransactionImpl`
Home Interface	`com.beasys.commerce.ebusiness.payment.PaymentTransactionHome`
Remote Interface	`com.beasys.commerce.ebusiness.payment.PaymentTransaction`
Type	Entity bean
Security Roles and Role Assignments	`CustomerRole` and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`CustomerRole`, `AdministrativeRole`

Table 4-28 Security Settings for the PriceService EJB

Name	PriceService
Description	Provides methods for pricing a shopping cart and order contents.

Table 4-28 Security Settings for the PriceService EJB (Continued)

Class	`com.bea.commerce.ebusiness.price.service.internal.PriceServiceImpl`
Home Interface	`com.bea.commerce.ebusiness.price.service.PriceServiceHome`
Remote Interface	`com.bea.commerce.ebusiness.price.service.PriceService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole` and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-29 Security Settings for the ProductItemManager EJB

Name	ProductItemManager
Description	Responsible for storing ProductItem and caching instances for the catalog.
Class	`com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManagerImpl`
Home Interface	`com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManagerHome`
Remote Interface	`com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `CustomerRole` can access all methods in the bean's Home interface, and the following methods on the bean's Remote interface: `getCatalogManager`, `getItemCount`, `getKeywords`, `getItemKeys`, `getItems`, `getItem`. `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AdministrativeRole`, `AnonymousRole`, `CustomerRole`

Table 4-30 Security Settings for the ShippingHelper EJB

Name	ShippingHelper
Description	Facilitates tasks related to management of shipping methods.
Class	`com.beasys.commerce.ebusiness.shipping.ShippingHelperImpl`
Home Interface	`com.beasys.commerce.ebusiness.shipping.ShippingHelperHome`
Remote Interface	`com.beasys.commerce.ebusiness.shipping.ShippingHelper`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-31 Security Settings for the ShippingMethod EJB

Name	ShippingMethod
Description	Encapsulates a delivery option, incorporating a standard set of attributes, and allows for complex shipping cost calculations.
Class	`com.beasys.commerce.ebusiness.shipping.ShippingMethodImpl`
Home Interface	`com.beasys.commerce.ebusiness.shipping.ShippingMethodHome`
Remote Interface	`com.beasys.commerce.ebusiness.shipping.ShippingMethod`
Type	Entity bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`

ejbadvisor.jar

The `ejbadvisor.jar` file contains an EJB that provides access to BEA Advisor personalization infrastructure. The `ejbadvisor.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- EjbAdvisor

Table 4-32 Security Settings for the EjbAdvisor EJB

Name	EjbAdvisor
Description	An EJB-based implementation of the BEA Advisor API. The Advisor API allows a caller to request general personalization advice. The Advisor implementation contains logic to interpret a request, sequence the appropriate personalization services, and return the resultant personalized advice to the caller.
Class	`com.bea.p13n.advisor.internal.EjbAdvisorImpl`
Home Interface	`com.bea.p13n.advisor.EjbAdvisorHome`
Remote Interface	`com.bea.p13n.advisor.EjbAdvisor`
Type	Stateless session bean
Security Roles and Role Assignments	`EveryoneRole` can access all methods in the bean's Home interface.
Security-Role References	None

events.jar

The `events.jar` file contains an EJB that supports the Events and Behavior Tracking service. The `events.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- EventService

Table 4-33 Security Settings for the EventService EJB

Name	EventService
Description	Responsible for dispatching events to registered listeners. Listeners register themselves for this service via the WebLogic Server Administration Console.
Class	`com.bea.p13n.events.internal.EventServiceBean`
Home Interface	`com.bea.p13n.events.EventServiceHome`
Remote Interface	`com.bea.p13n.events.EventService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	None

ldapprofile.jar

The `ldapprofile.jar` file contains an EJB that is used to read user and group properties from an LDAP server. The `ldapprofile.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- LdapPropertyManager

Table 4-34 Security Settings for the LdapPropertyManager EJB

Name	LdapPropertyManager
Description	An implementation of EntityPropertyManager that retrieves property data from an LDAP server. Connection information for the LDAP server is contained in environment entries in the deployment desciptor.

Table 4-34 Security Settings for the LdapPropertyManager EJB

Class	`com.bea.p13n.usermgmt.profile.ldap.internal.LdapPropertyManagerImpl`
Home Interface	`com.bea.p13n.usermgmt.profile.ldap.LdapPropertyManagerHome`
Remote Interface	`com.bea.p13n.usermgmt.profile.ldap.LdapPropertyManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

mail.jar

The `mail.jar` file is a collection of EJBs that gives your application access to the outbound Mail Service, which uses the JavaMail API to send e-mail to customers in batches. The `mail.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- MailService

Table 4-35 Security Settings for the MailService EJB

Name	MailService
Description	Provides services for batching and sending e-mail messages.
Class	`com.bea.p13n.mail.internal.MailServiceImpl`
Home Interface	`com.bea.p13n.mail.MailServiceHome`
Remote Interface	`com.bea.p13n.mail.MailService`
Type	Stateless session bean

Table 4-35 Security Settings for the MailService EJB (Continued)

Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `sendMail`, `addToBatch`, `getTextFromJSP`, `getJSPResults`. `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

payment.jar

The `payment.jar` file contains an EJB that supports the Payment Web Service application. The `payment.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- CreditCardWebService

Table 4-36 Security Settings for the CreditCardWebService EJB

Name	CreditCardWebService
Description	Implementation of the sample Payment Web Service used by the Commerce templates during order checkout.
Class	`examples.wlcs.sampleapp.payment.CreditCardWebServiceImpl`
Home Interface	`examples.wlcs.sampleapp.payment.CreditCardWebServiceHome`
Remote Interface	`examples.wlcs.sampleapp.payment.CreditCardWebService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole` and `AdministrativeRole` can access all the methods in the bean's Home and Remote interfaces.
Security-Role References	None

petflow.jar

The `petflow.jar` file contains an EJB and all the Pipeline Components for the Pet Flow sample Web application.

Note: All the Pipeline Components are application scoped, must be deployed with an EJB, and not reside in the system classpath or Web application classpath. Alternately, if the Pipeline Components are not EJBs they may be packaged in a separate JAR (as long as they are referenced from the `pipeline.jar’s` manifest file). Since the components in the `petflow.jar` file reference components in other utility JARs, you must update the classpath in `petflow.jar’s` manifest file to include the JARs of the utility components: `Class-Path: p13n_util.jar petStore_EJB.jar pipeline.jar.`

The `petflow.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- GetInventoryPC

Table 4-37 Security Settings for the GetInventoryPC EJB

Name	GetInventoryPC
Description	Retrieves the inventory (number of items in stock) for all the items in the shopping cart.
Class	`examples.petflow.pipeline.ejb.GetInventoryPCImpl`
Home Interface	`examples.petflow.pipeline.ejb.GetInventoryPCHome`
Remote Interface	`examples.petflow.pipeline.ejb.GetInventoryPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean’s Home and Remote interfaces.
Security-Role References	None

petStore_EJB.jar

The `petStore_EJB.jar` file is a collection of EJBs and other application-scoped classes that are part of the Sun Microsystems Pet Store Demo, a comprehensive e-commerce application based on the Java 2 Enterprise Edition (J2EE) specification. For more information on the original Pet Store Demo, see the "Architecture Overview" at http://java.sun.com/j2ee/blueprints/jps11/archoverview.html.

The EJBs and other classes in this JAR file are referenced by the Pet Flow sample Web application, which comes packaged with the WebLogic Portal product suite. The Pet Flow application has been designed to illustrate some of the features of the Webflow mechanism.

pipeline.jar

The `pipeline.jar` file contains all Pipeline internals, including an EJB that provides support for the Pipeline service and other supporting, application-scoped classes.

Note: This file need only be deployed if Pipeline Components are being used. If you are strictly using only Webflow you need not deploy this file.

The `pipeline.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- PipelineExecutor

Table 4-38 Security Settings for the PipelineExecutor EJB

Name	PipelineExecutor
Description	The PipelineProcessor (`com.bea.p13n.appflow.webflow.internal.PipelineProcessor`) is an extension (custom) processor that delegates to the PipelineExecutor stateless session EJB. It is the PipelineExecutor that parses the Pipeline XML, invokes the Pipeline's components, and demarcates all transactions within the Pipeline and Pipeline Session.

Table 4-38 Security Settings for the PipelineExecutor EJB

Class	`com.bea.p13n.appflow.pipeline.internal.PipelineExecutorImpl`
Home Interface	`com.bea.p13n.appflow.pipeline.PipelineExecutorHome`
Remote Interface	`com.bea.p13n.appflow.pipeline.PipelineExecutor`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	None

placeholder.jar

The `placeholder.jar` file is a collection of EJBs that support the placeholder, ads, and ad bucket services. The `placeholder.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- AdBucketService
- AdConflictResolver
- AdService
- PlaceholderService

Table 4-39 Security Settings for the AdBucketService EJB

Name	AdBucketService
Description	Ties into the PlaceholderService EJB to provide per-user, per-placeholders buckets of ad queries.
Class	`com.bea.p13n.ad.internal.AdBucketServiceBean`
Home Interface	`com.bea.p13n.ad.AdBucketServiceHome`

Table 4-39 Security Settings for the AdBucketService EJB

Remote Interface	`com.bea.p13n.ad.AdBucketService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface and the following methods in the bean's Remote interface: `getContent`, `userAddAd`, `userClearAds`, `getAdEventInfo`, `previewContent`, `removeGlobalQueries`, and `setGlobalQuery`. `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-40 Security Settings for the AdConflictResolver EJB

Name	AdConflictResolver
Description	Resolves which of a set of ad queries should be used to display an ad to a user.
Class	`com.bea.p13n.ad.internal.AdConflictResolverBean`
Home Interface	`com.bea.p13n.ad.AdConflictResolverHome`
Remote Interface	`com.bea.p13n.ad.AdConflictResolver`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-41 Security Settings for the AdService EJB

Name	AdService
Description	Handles retrieving ads and managing clicks and impressions.

Table 4-41 Security Settings for the AdService EJB

Class	`com.bea.p13n.ad.internal.AdServiceBean`
Home Interface	`com.bea.p13n.ad.AdServiceHome`
Remote Interface	`com.bea.p13n.ad.AdService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

Table 4-42 Security Settings for the PlaceholderService EJB

Name	PlaceholderService
Description	Retrieves displayable content for a placeholder for a user.
Class	`com.bea.p13n.placeholder.internal.PlaceholderServiceImpl`
Home Interface	`com.bea.p13n.placeholder.PlaceholderServiceHome`
Remote Interface	`com.bea.p13n.placeholder.PlaceholderService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface and the following methods in the bean's Remote interface: `getContent`, `previewContent`, `getPreviewSlot`, and `removePreviewSlot`. `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

portal.jar

The `portal.jar` file is a collection of EJBs that provide all portal functionality, including the visitor runtime tools and the administration tools. The `portal.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- AdminResourceManager
- DelegatedAdminManager
- PortalManager
- VisitorUserManager

Table 4-43 Security Settings for the AdminResourceManager EJB

Name	AdminResourceManager
Description	Administration EJBs that allow `SystemAdministrators` and `DelegatedAdministrators` to perform administrative functions on portals and group portals to which they have access.
Class	`com.bea.portal.admin.ejb.internal.AdminResourceManagerBean`
Home Interface	`com.bea.portal.admin.ejb.AdminResourceManagerHome`
Remote Interface	`com.bea.portal.admin.ejb.AdminResourceManager`
Type	Stateless session bean
Security Roles and Role Assignments	`DelegatedAdminRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

Table 4-44 Security Settings for the DelegatedAdminManager EJB

Name	DelegatedAdminManager
Description	Administration EJBs that manage the activities associated with `DelegatedAdministrators`, including creating administrators, retrieving administrators, deleting administrators, promoting administrators, updating administrator privileges, and so on.
Class	`com.bea.portal.admin.ejb.internal.DelegatedAdminManagerBean`
Home Interface	`com.bea.portal.admin.ejb.DelegatedAdminManagerHome`
Remote Interface	`com.bea.portal.admin.ejb.DelegatedAdminManager`
Type	Stateless session bean
Security Roles and Role Assignments	`DelegatedAdminRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

Table 4-45 Security Settings for the PortalManager EJB

Name	PortalManager
Description	Visitor (not administration) EJB that handles all runtime rendering of portlets as well as personalization information for a visitor's portal. This rendering includes skins, layouts, displayed portlets, and so on.
Class	`com.bea.portal.manager.ejb.internal.PortalManagerBean`
Home Interface	`com.bea.portal.manager.ejb.PortalManagerHome`
Remote Interface	`com.bea.portal.manager.ejb.PortalManager`
Type	Stateless session bean
Security Roles and Role Assignments	`EveryoneRole` can access all methods in the bean's Home interface.

Table 4-45 Security Settings for the PortalManager EJB

Security-Role References	`EveryoneRole`

Table 4-46 Security Settings for the VisitorUserManager EJB

Name	VisitorUserManager
Description	Administrator EJB that manages visitor users. Most User Management and Group Management EJB methods (from `usermgmt.jar`) are exposed and wrapped with a portal-specific security layer for `DelegatedAdministrators`.
Class	`com.bea.portal.admin.ejb.internal.VisitorUserManagerBean`
Home Interface	`com.bea.portal.admin.ejb.VisitorUserManagerHome`
Remote Interface	`com.bea.portal.admin.ejb.VisitorUserManager`
Type	Stateless session bean
Security Roles and Role Assignments	`DelegatedAdminRole` and `SystemAdminRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

property.jar

The `property.jar` file is a collection of EJBs that provide Property Set functionality and persistence services for components that use the property framework, such as User and Group profiles. The `property.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- EntityPropertyManager
- PropertySetManager

Table 4-47 Security Settings for the EntityPropertyManager EJB

Name	EntityPropertyManager
Description	EJB that provides persistence services for entity EJBs that extend ConfigurableEntity, and for User and Group profiles.
Class	`com.bea.p13n.property.internal.EntityPropertyManagerImpl`
Home Interface	`com.bea.p13n.property.EntityPropertyManagerHome`
Remote Interface	`com.bea.p13n.property.EntityPropertyManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	None

Table 4-48 Security Settings for the PropertySetManager EJB

Name	PropertySetManager
Description	EJB that handles the deployment and runtime management of property sets.
Class	`com.bea.p13n.property.internal.PropertySetManagerImpl`
Home Interface	`com.bea.p13n.property.PropertySetManagerHome`
Remote Interface	`com.bea.p13n.property.PropertySetManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	None

propertysetws.jar

The `propertysetws.jar` file contains an EJB (that provides a Web service) that provides access to server side Property Sets so that outside components, such as the E-Business Control Center, can use them. The `propertysetws.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- PropertySetWebService

Table 4-49 Security Settings for the PropertySetWebService EJB

Name	PropertySetWebService
Description	EJB wrapped by a Web service that gives access to property sets via SOAP calls.
Class	`com.bea.p13n.property.webservice.internal.PropertySetWebServiceImpl`
Home Interface	`com.bea.p13n.property.webservice.PropertySetWebServiceHome`
Remote Interface	`com.bea.p13n.property.webservice.PropertySetWebService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

rules.jar

The `rules.jar` file contains an EJB that provides stateless access to the underlying BEA Rules Engine. The `rules.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- RulesManager

Table 4-50 Security Settings for the RulesManager EJB

Name	RulesManager
Description	Exposes an API to the underlying BEA Rules Engine that allows a caller to execute rule sets and/or named rules. The RulesManager also performs BEA Rules Engine resource caching and pooling in order to improve rule execution performance.
Class	`com.bea.p13n.rules.manager.internal.RulesManagerImpl`
Home Interface	`com.bea.p13n.rules.manager.RulesManagerHome`
Remote Interface	`com.bea.p13n.rules.manager.RulesManager`
Type	Stateless session bean
Security Roles and Role Assignments	`EveryoneRole` can access all methods in the bean's Home interface, and the following methods in the bean's Remote interface: `evaluateRuleSet` and `evaluateRule`. `RulesReaderRole` can access the `getRuleSet` method in the bean's Remote interface. `SystemAdminRole` can access the following methods in the bean's Remote interface: `setRuleSet`, `getRuleSet`, `getRuleSets`, and `removeRuleSet`.
Security-Role References	None

stockportal.jar

The `stockportal.jar` file contains the code and a Pipeline Component for the Worldnews portlet. The `stockportal.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- moreoverNewsPC

Table 4-51 Security Settings for the moreoverNewsPC EJB

Name	moreoverNewsPC

Table 4-51 Security Settings for the moreoverNewsPC EJB

Description	A Pipeline Component EJB that can retrieve news article information from moreover.com.
Class	`examples.worldnews.pipeline.ejb.internal.MoreoverNewsPCBean`
Home Interface	`examples.worldnews.pipeline.ejb.MoreoverNewsPCHome`
Remote Interface	`examples.worldnews.pipeline.ejb.MoreoverNewsPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole` and `AdministrativeRole` can access all the methods in the bean's Home and Remote interfaces.
Security-Role References	None

tax.jar

The `tax.jar` file contains an EJB that supports the Tax Web Service application. The `tax.jar` file contains the following EJB, whose `ejb-jar.xml` deployment descriptor security settings are described in the table that follows:

- TaxWebService

Table 4-52 Security Settings for the TaxWebService EJB

Name	TaxWebService
Description	Implementation of the sample Tax Web Service used by the Commerce templates during order checkout.
Class	`examples.wlcs.sampleapp.tax.TaxWebServiceImpl`
Home Interface	`examples.wlcs.sampleapp.tax.TaxWebServiceHome`
Remote Interface	`examples.wlcs.sampleapp.tax.TaxWebService`
Type	Stateless session bean

Table 4-52 Security Settings for the TaxWebService EJB

Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	None

usermgmt.jar

The `usermgmt.jar` file is a collection of EJBs that provide user and group management and Unified User Profile (UUP) services. The `usermgmt.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- GroupProfileManager
- GroupManager
- RealmConfiguration
- UserProfileManager
- UserProfileManager

Table 4-53 Security Settings for the GroupProfileManager EJB

Name	GroupProfileManager
Description	Manages property access for group profiles.
Class	`com.bea.p13n.usermgmt.profile.internal. GroupProfileManagerImpl`
Home Interface	`com.bea.p13n.usermgmt.profile.ProfileManagerHome`
Remote Interface	`com.bea.p13n.usermgmt.profile.ProfileManager`
Type	Stateless session bean

Table 4-53 Security Settings for the GroupProfileManager EJB (Continued)

Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

Table 4-54 Security Settings for the GroupManager EJB

Name	GroupManager
Description	Provides group management services, such as creating and deleting groups, and manipulating group memberships.
Class	`com.bea.p13n.usermgmt.internal.GroupManagerImpl`
Home Interface	`com.bea.p13n.usermgmt.GroupManagerHome`
Remote Interface	`com.bea.p13n.usermgmt.GroupManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

Table 4-55 Security Settings for the RealmConfiguration EJB

Name	RealmConfiguration
Description	Helps keep security and profile information synchronized. This is used only by the administration tools.
Class	`com.bea.p13n.usermgmt.config.internal.RealmConfigurationImpl`
Home Interface	`com.bea.p13n.usermgmt.config.RealmConfigurationHome`
Remote Interface	`com.bea.p13n.usermgmt.config.RealmConfiguration`

Table 4-55 Security Settings for the RealmConfiguration EJB

Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	None

Table 4-56 Security Settings for the UserManager EJB

Name	UserManager
Description	Provides user management services, such as creating and deleting users, and managing user profile types.
Class	`com.bea.p13n.usermgmt.internal.UserManagerImpl`
Home Interface	`com.bea.p13n.usermgmt.UserManagerHome`
Remote Interface	`com.bea.p13n.usermgmt.UserManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

Table 4-57 Security Settings for the UserProfileManager EJB

Name	UserProfileManager
Description	Manages property access for user profiles.
Class	`com.bea.p13n.usermgmt.profile.internal.UserProfileManagerImpl`
Home Interface	`com.bea.p13n.usermgmt.profile.ProfileManagerHome`

Table 4-57 Security Settings for the UserProfileManager EJB

Remote Interface	`com.bea.p13n.usermgmt.profile.ProfileManager`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home interface.
Security-Role References	`DelegatedAdminRole` and `SystemAdminRole`

wlcsSample.jar

The `wlcsSample.jar` file is a collection of EJBs that provide various components used exclusively by the Commerce JSP templates. The `wlcsSample.jar` file contains the following EJBs, whose `ejb-jar.xml` deployment descriptor security settings are described in the tables that follow:

- CreditCardService
- DeleteProductItemFromSavedListNewPC
- MoveProductItemToSavedListNewPC
- MoveProductItemToShoppingCartNewPC
- RefreshSavedListNewPC
- TaxCalculator

Table 4-58 Security Settings for the CreditCardService EJB

Name	CreditCardService
Description	The gateway or wrapper to back-end Payment Services. Also responsible for miscellaneous business tasks like payment transaction journaling.
Class	`examples.wlcs.sampleapp.payment.CreditCardServiceImpl`
Home Interface	`examples.wlcs.sampleapp.payment.CreditCardServiceHome`

Table 4-58 Security Settings for the CreditCardService EJB

Remote Interface	`examples.wlcs.sampleapp.payment.CreditCardService`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole`, `CustomerRole`, and `AdministrativeRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`CustomerRole`, `AdministrativeRole`

Table 4-59 Security Settings for DeleteProductItemFromSavedListNewPC EJB

Name	DeleteProductItemFromSavedListNewPC
Description	Pipeline Component that removes a product item from the list of saved items for a given customer.
Class	`examples.wlcs.sampleapp.shoppingcart.pipeline.DeleteProductItemFromSavedListPCImpl`
Home Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.DeleteProductItemFromSavedListPCHome`
Remote Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.DeleteProductItemFromSavedListPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`

Table 4-60 Security Settings for the MoveProductItemToSavedListNewPC EJB

Name	MoveProductItemToSavedListNewPC
Description	Pipeline Component that removes a product item from a customer's shopping cart and adds it to their list of saved items.

Table 4-60 Security Settings for the MoveProductItemToSavedListNewPC EJB (Continued)

Class	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToSavedListPCImpl`
Home Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToSavedListPCHome`
Remote Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToSavedListPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean’s Home and Remote interfaces.
Security-Role References	`AnonymousRole`

Table 4-61 Security Settings for MoveProductItemToShoppingCartNewPC EJB

Name	MoveProductItemToShoppingCartNewPC
Description	Pipeline Component that removes a product item from a customer's list of saved items and adds it to their shopping cart.
Class	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToShoppingCartPCImpl`
Home Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToShoppingCartPCHome`
Remote Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.` `MoveProductItemToShoppingCartPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean’s Home and Remote interfaces.
Security-Role References	`AnonymousRole`

Table 4-62 Security Settings for the RefreshSavedListNewPC EJB

Name	RefreshSavedListNewPC
Description	Pipeline Component that updates a customer's list of saved items in the Pipeline Session.
Class	`examples.wlcs.sampleapp.shoppingcart.pipeline.RefreshSavedListPCImpl`
Home Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.RefreshSavedListPCHome`
Remote Interface	`examples.wlcs.sampleapp.shoppingcart.pipeline.RefreshSavedListPC`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`

Table 4-63 Security Settings for the TaxCalculator EJB

Name	TaxCalculator
Description	The gateway or wrapper to back-end Tax Services.
Class	`examples.wlcs.sampleapp.tax.TaxCalculatorImpl`
Home Interface	`examples.wlcs.sampleapp.tax.TaxCalculatorHome`
Remote Interface	`examples.wlcs.sampleapp.tax.TaxCalculator`
Type	Stateless session bean
Security Roles and Role Assignments	`AnonymousRole` can access all methods in the bean's Home and Remote interfaces.
Security-Role References	`AnonymousRole`

CHAPTER

5 Security in the WebLogic Portal Administration Tools and the E-Business Control Center

In addition to the sample applications, WebLogic Portal also includes some applications that you may use to develop and maintain your e-commerce Web site. Presently, these tools are available in two forms: the browser-based WebLogic Portal Administration Tools and the stand-alone E-Business Control Center.

The WebLogic Portal Administration Tools and the E-Business Control Center (EBCC) are designed for use by the specific individuals within your organization who are authorized to make modifications to your e-business Web site. Thus, both applications require some security protections. The security implemented in these applications are discussed in the following sections:

- Security in the WebLogic Portal Administration Tools
- Security in the E-Business Control Center

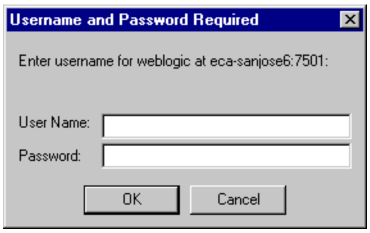
Security in the WebLogic Portal Administration Tools

The purpose of the browser-based WebLogic Portal Administration Tools is to provide administrative users of WebLogic Portal with the ability to manage various aspects of an e-commerce Web application. The WebLogic Portal Administration Tools application includes JSPs for Portal Management, User Management, Catalog Management, Order Management, and Payment Management. Thus, access to these administration JSPs must be granted only to authorized users.

Note: For detailed information about accessing the WebLogic Portal Administration Tools, logging in, or changing the Administrator password, see “WebLogic Portal Administration Tools” in the *WebLogic Portal Architectural Overview*.

When users attempt to access the WebLogic Portal Administration Tools, the application invokes basic authentication techniques to present a login screen to the user. An example of a login screen is shown in Figure 5-1.

Figure 5-1 Administration Login Screen

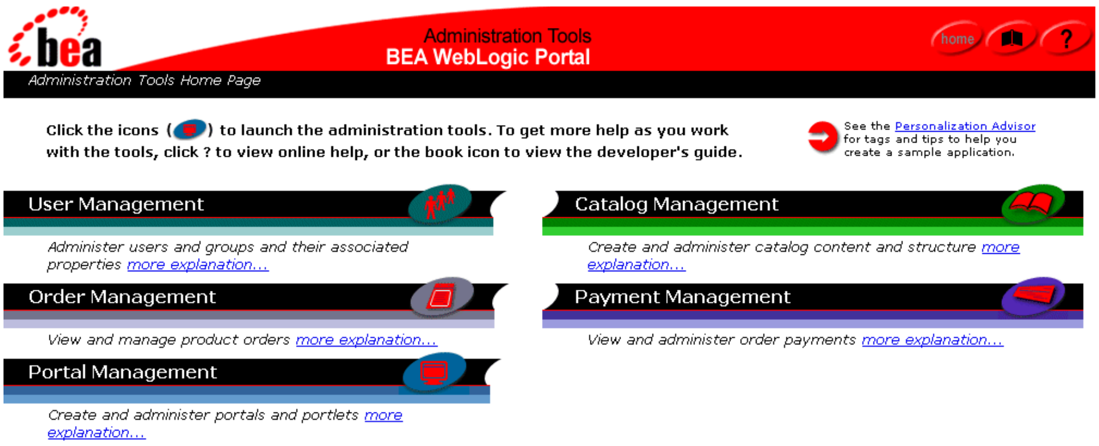

Username and Password Required
Enter username for weblogic at eca-sanjose6:7501:
User Name:
Password:
OK
Cancel

Note: With basic authentication, the WebLogic Server instructs the Web client to prompt for a username and password, which the server then uses to authenticate a principal. For more information about authentication and principals, see “Users and User Groups as Principals” on page 2-4.

When the user submits their username and password to gain access to the WebLogic Portal Administration Tools, the submitted information is passed to the WebLogic Server for verification. The security mechanism in WebLogic Server verifies that the user is part of the `SystemAdministrator` or `DelegatedAdministrator` user group and that the user has supplied the correct password. If the user supplies a valid username and password combination, WebLogic Server authenticates the user and grants access to the WebLogic Portal Administration Tools.

Presently, users who have been successfully authenticated by the WebLogic Server and belong to the `SystemAdministrator` user group have permission to use all the features available in the WebLogic Portal Administration Tools. As such, the WebLogic Portal Administration Tools home page will be presented to these users following authentication (Figure 5-2).

Figure 5-2 WebLogic Portal Administration Tools Home Page



Click the icons ( ) to launch the administration tools. To get more help as you work with the tools, click ? to view online help, or the book icon to view the developer's guide.

See the Personalization Advisor for tags and tips to help you create a sample application.

User Management

Administer users and groups and their associated properties more explanation...

Catalog Management

Create and administer catalog content and structure more explanation...

Order Management

View and manage product orders more explanation...

Payment Management

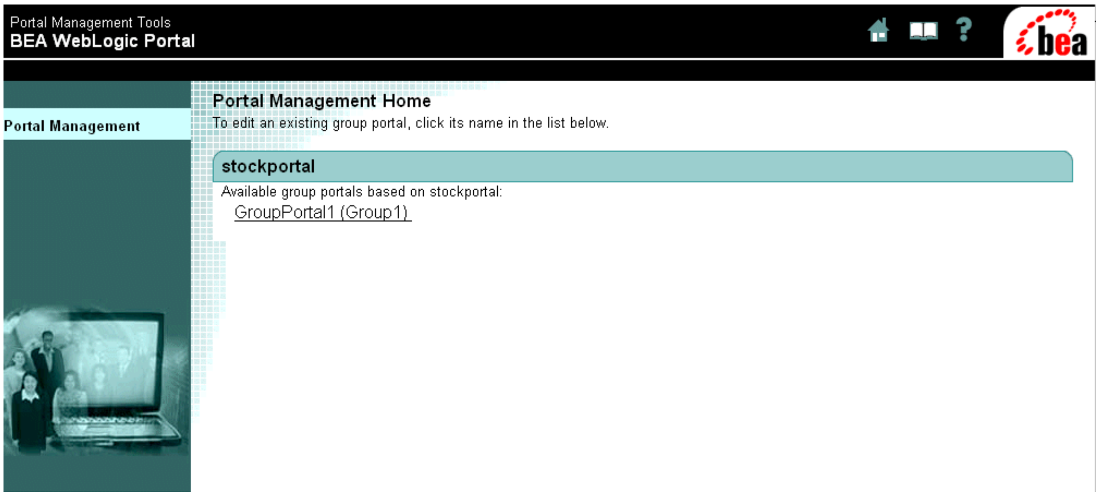
View and administer order payments more explanation...

Portal Management

Create and administer portals and portlets more explanation...

Alternatively, users who have been successfully authenticated and belong to the `DelegatedAdministrator` user group have permission to use the Portal Management features only. Therefore, these users will be presented with the Portal Management home page following authentication (Figure 5-3). Additionally, the Portal Management home page will display only the portals or group portals to which the user has access (depending on whether the `DelegatedAdministrator` is further defined as a Portal Administrator (PA) or a Group Administrator (GA)). For more information, see Chapter 6, "Portal Administration and Security."

Figure 5-3 Portal Management Home Page



Portal Management Home

To edit an existing group portal, click its name in the list below.

stockportal

Available group portals based on stockportal:

GroupPortal1 (Group1)

Security in the E-Business Control Center

The E-Business Control Center (EBCC) is an application available for use with WebLogic Portal. The E-Business Control Center is designed to simplify the tasks that are necessary to create and maintain a truly personalized Web site. To meet this objective, the E-Business Control Center guides both business and technical users through a variety of tasks, ensuring that people in these diverse roles can focus on the aspects of e-business management that are relevant to them.

The E-Business Control Center is similar to the WebLogic Portal Administration Tools in that it allows certain, privileged users to affect the content and behavior of a Web site. However, the similarities end there. The E-Business Control Center:

- Is a stand-alone GUI application (rather than a set of Web-based tools built using JavaServer Pages) that is not started within a Web browser.
- Allows authorized users to edit development-time aspects of a Web site, rather than the administrative (runtime) aspects (which are modified using the WebLogic Portal Administration Tools).

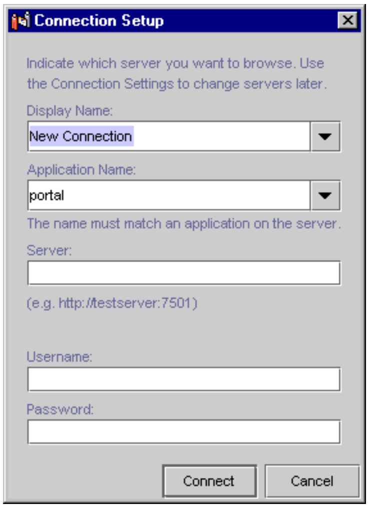
Notes: For information about installing the E-Business Control Center, see “Installing the E-Business Control Center” in the *Installation Guide*. For instructions on how to access the E-Business Control Center, see “Starting the E-Business Control Center” in the *Guide to Using the E-Business Control Center* documentation.

The E-Business Control Center works against files, so users of this tool can view and modify any files that reside locally on their filesystem. Security is required, however, when the E-Business Control Center communicates with a WebLogic Portal server via servlet calls. Some of these servlets require basic authentication to be performed before they can be accessed because they are protected by standard Web application security mechanisms. Therefore, although the majority of E-Business Control Center functionality does not require users to log in to a running WebLogic Portal server, users are required to log in when viewing or using certain data that causes the protected servlets to be called. Users must log in to the server to:

- Search for or preview ads in various windows.
- Access property sets in various windows.
- Use the Catalog Browser in various windows to select items in a catalog.
- Select and preview e-mail content that will be sent out automatically as part of a promotional campaign.
- Synchronize your application data to the server.

In these cases, a user is first prompted to connect to a server by providing some connection information, which includes the name of the WebLogic Portal server, a username and a password. This information is typically gathered in the Connections Setup window, shown in Figure 5-4.

Figure 5-4 Connection Setup Window



Connection Setup

Indicate which server you want to browse. Use the Connection Settings to change servers later.

Display Name: New Connection

Application Name: portal

The name must match an application on the server.

Server:

(e.g. http://testserver:7501)

Username:

Password:

Notes: If prompted to login prior to a data synchronization operation, this information is gathered in the Connections tab of the Synchronization Setup window. For details about these windows and the information required to connect, see "Connecting to the Server" in the *Guide to Using the E-Business Control Center* documentation.

If you are using the E-Business Control Center to make server-encrypted connections using SSL (Secure Socket Layer), then the information you specify in the Server input field should start with `https://` Remember that for SSL connections to work, you must have a valid SSL certificate from a certificate authority set up on your server. For more information about certificates, see "Digital Certificates" in the *Programming WebLogic Server Security* documentation.

The username/password combination required for a data synchronization operation (and all other protected operations) is either:

- The `system` user only (and not all members of the `Administrator` user group).
- Any user in the `SystemAdministrator` user group.

Note: The username/password combination is authorized by the WebLogic Server's security mechanisms.

Note: The features available to authorized users of the E-Business Control Center depend on the version of the application present in your organization. Versions of the E-Business Control Center are determined by product license. For more information on application versions, see “What the E-Business Control Center Provides” in the “Introduction” topic of the *Guide to Using the E-Business Control Center* documentation.

6 Portal Administration and Security

Managing WebLogic Portal effectively requires an understanding of J2EE security concepts as well as a grasp of security features unique to the WebLogic Portal platform. Portal administration requires the ability to manage access on a many-to-many basis. In other words, access to many different groupings of resources must be provided to many different groupings of users. For this reason, the WebLogic Portal platform goes beyond the J2EE security standard as currently written, and also encompasses existing WebLogic Server security schemes.

This topic provides additional detail about the three levels of administration that WebLogic Portal supports, and includes information about how the access granted to these administrator users is scoped. This topic also includes information about how administrative users are managed, and how administration tasks may be delegated. Finally, some information about visitor entitlements for WebLogic Portal is provided.

This topic includes the following sections:

- Three Levels of Administrator Permissions
  - SA- System Administrators
  - PA - Portal Administrators
  - GA - Group Administrators
  - Application Assembler/Deployer
  - Scoping Privileges
- Managing Administrator Users
  - User Groups

- Delegated Administration

■ Visitor Entitlements

- Rule-Based Entitlements Versus Rule-Based Personalization

Note: For more information about how to manage the WebLogic Portal platform effectively by performing specific administration tasks, see “Overview of Portal Administration” in the *Getting Started with Portals and Portlets* documentation.

Three Levels of Administrator Permissions

The WebLogic Portal platform recognizes three basic subdivisions of Administrators: the System Administrator (Portal SA), Portal Administrator (Portal PA), and Group Administrator (Portal GA). Individual Portal PAs and Portal SAs can be assigned fine-grained privileges, enabling the creation of a very complex administrator hierarchy customized to fit very specific security models.

Note: A list of out-of-the-box users can be found in “Overview of Portal Administration.”

It is important to note the distinction between these Administrators and `system`, the default account used to control the WebLogic Server Administration Console. The Portal SAs, PAs and GAs use the browser-based WebLogic Portal Administration Tools, whereas `system` is the sole member of a special, unchangeable user group called `Administrator` used to start and stop WebLogic Server.

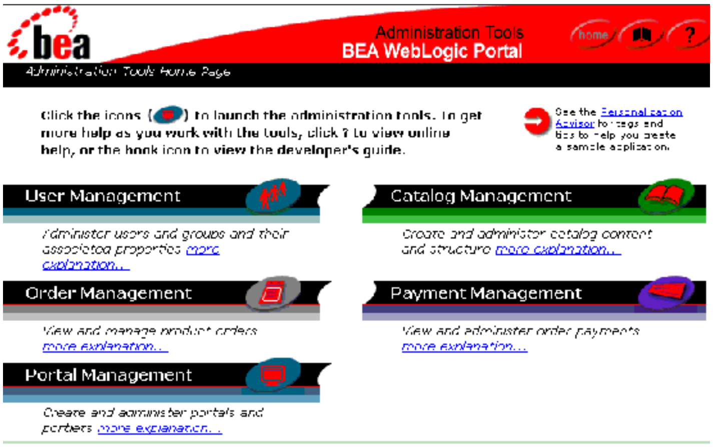
Note: For instructions on creating, editing, and deleting Portal SA, PA, and GA users, see “Portal Administration Tools” in the *Getting Started with Portals and Portlets* documentation.

SA- System Administrators

Out of the box, WebLogic Portal includes a Portal SA user called `administrator`, which has unlimited access to administrative tasks anywhere within the enterprise portal application.

When a Portal SA logs into portalTools Web application, (by navigating to `http://<hostname>:<port>/portalTools/index.jsp`), the WebLogic Portal Administration Tools page appears, as shown in Figure 6-1.

Figure 6-1 WebLogic Portal Administration Tools Home Page



Administration Tools
BEA WebLogic Portal
Administration Tools Home Page

Click the icons ( ) to launch the administration tools. To get more help as you work with the tools, click ? to view online help, or the book icon to view the developer's guide.

See the Personalization Advisor for tags and tips to help you create a sample application.

User Management
Administer users and groups and their associated properties more explanation...

Catalog Management
Create and administer catalog content and structure more explanation...

Order Management
View and manage product orders more explanation...

Payment Management
View and administer order payments more explanation...

Portal Management
Create and administer portals and portlets more explanation...

Within the WebLogic Portal, the System Administrator (Portal SA) may perform any of the following administrative actions:

- Deploy portal applications using the E-Business Control Center
- Create a new user group
- Create and entitle Portal Administrators (PAs)
- Create/Edit a group portal
- Create and entitle Group Administrators (GAs)
- Delegate administrative tasks

Because the Portal SA has access to all possible administrative tasks available within the WebLogic Portal Administration Tools, it is recommended that you observe the following guidelines:

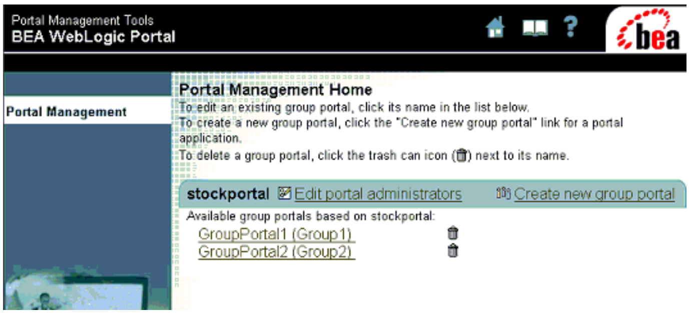
- Only use `system` to create new users, place new users in the `SystemAdministrator` user group (thereby making them Portal SAs), adding users to user groups such as `AdminEligible`, or to create a new user group. Limit the use of this powerful user, and remember to log off when necessary tasks have been completed. This will lessen the chance of its credential being compromised.
- Limit the number of users you place in the `SystemAdministrator` user group. Fewer Portal SA users on your instance of WebLogic Portal will reduce the effort required to keep the passwords fresh, and lessen the probability that one of these user accounts may be compromised.

PA - Portal Administrators

Out of the box, WebLogic Portal includes a pair of test users called `demopa1` and `demopa2`. These users have access to administrative tasks on a portal-wide basis, and as such, they can be be granted administrative priveleges to any group portal within the portal Web application.

When a Portal PA logs into portalTools Web application, (by navigating to `http://<hostname>:<port>/portalTools/index.jsp`), the WebLogic Portal Management home page appears, as shown in Figure 6-2.

Figure 6-2 WebLogic Portal Management Home Page



Portal Management Home

To edit an existing group portal, click its name in the list below.
To create a new group portal, click the "Create new group portal" link for a portal application.
To delete a group portal, click the trash can icon (🗑) next to its name.

stockportal Edit portal administrators Create new group portal

Available group portals based on stockportal:
GroupPortal1 (Group1)
GroupPortal2 (Group2)

Within one or more portal Web applications, the Portal Administrator (PA) may be granted the ability to perform any of the following administrative actions:

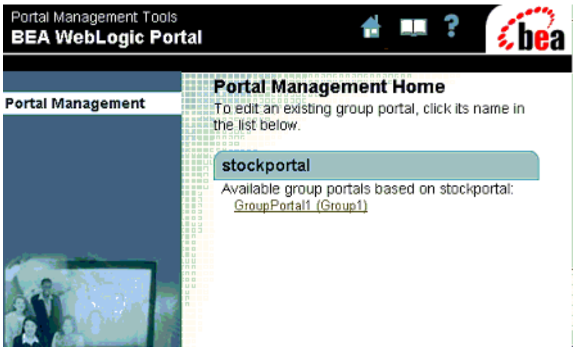
- Create and entitle Portal Administrators (PAs)
- Create/Edit a group portal
- Create and entitle Group Administrators (GAs)
- Delegate administrative tasks

GA - Group Administrators

Out of the box, WebLogic Portal includes a pair of test users called `demoga1` and `demoga2`. These users have access to administrative tasks at the group portal level. These users can be granted administrative priveleges to any group portal within the portal Web application. Also, Portal GAs can be promoted by Portal SA or Portal PA users.

When a Portal GA logs into portalTools Web application, (by navigating to `http://<hostname>:<port>/portalTools/index.jsp`), the WebLogic Group Portal Management home page appears, as shown in Figure 6-3.

Figure 6-3 WebLogic Group Portal Management Home Page



Portal Management Home

To edit an existing group portal, click its name in the list below.

stockportal

Available group portals based on stockportal:
GroupPortal1 (Group1)

Note: Portal GA users may manage more than one group portal. In these cases, the page shown in Figure 6-3 would include the name of another group portal.

Within one or more group portals, the Portal Administrator (Portal PA) may be granted the ability to perform any of the following administrative actions:

- Create and entitle Group Administrators (GAs)
- Edit a group portal
- Delegate administrative tasks

Application Assembler/Deployer

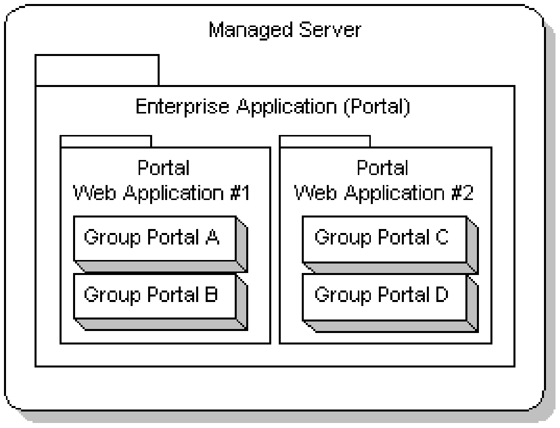
Though not given a specific role within the administrative workflow of WebLogic Portal, the Application Assembler/Deployer is distinct in that it is most closely associated with the use of the Portal Module of the E-Business Control Center.

The Application Deployer/Assembler also performs the synchronize task using the E-Business Control Center, which requires Portal SA privileges. (In other words, the Application Deployer/Assembler must be a member of the `SystemAdministrator` user group.)

Scoping Privileges

J2EE application scoping is the basis for the three levels of administration in WebLogic Portal. The scope of each of these administrators can be explained in terms of the application scoping shown in Figure 6-4.

Figure 6-4 Scoping of Administrators


Managed Server
Enterprise Application (Portal)
Portal Web Application #1
Group Portal A
Group Portal B
Portal Web Application #2
Group Portal C
Group Portal D

- A Portal SA created within the enterprise application called Portal could perform all administrative tasks within any of the applications shown in Figure 6-4.
- A Portal PA created within Portal Web Application #1 could be granted privileges within that application only. (Portal PAs can be granted different sets of privileges within more than one portal Web application.)
- A Portal GA created within Group Portal C could be granted privileges within that group portal only. (Portal GAs can be granted different sets of privileges within more than one group portal.)

Managing Administrator Users

The User Management home page allows Portal SA users to add or remove specific users from pre-defined user groups. After the user belongs to the appropriate user group, a mechanism called Delegated Administration is used to bestow specific privileges upon a single user. Therefore, this section includes information about:

- User Groups
- Delegated Administration

User Groups

The relationship between WebLogic Portal user groups and what membership in these user groups grants to the user is as follows.

- When any user is added to WebLogic Portal, the user is automatically a member of a user group called `everyone`.
- When a company creates user accounts for administrators, all these users should be added to the `AdminEligible` user group.
- Any user can be turned into an all-powerful system administrator— regardless of membership in other user groups: simply add the user to the `SystemAdministrator` user group from the User Management home page. Adding or removing a user from this group does not affect any other user group memberships.
- Importantly, promoting a user to GA or PA within the Portal Management or Group Portal Management home pages does eliminate other group memberships for that user. For example, if a GA user is promoted to PA, that user's membership as a GA is eliminated.

Delegated Administration

Delegated administration means assigning specific administrative privileges to individual administrators within a specified domain.

To manage the content of your portal applications at a centralized corporate office while delegating localization and design to regional offices, an elegant solution would be to create some administrative user accounts and then assign different sets of permissions to each user, based on the tasks you needed to hand out. WebLogic Portal now includes advanced delegated administration functionality to enable the creation of administrator roles with fine-grained administrative privileges.

Visitor Entitlements

Administrators are not the only kind of “user” that must be managed in a portal Web application; WebLogic Portal allows content to be customized for customers and business partners, called portal visitors. **Visitor entitlements** are associations between portal resources, such as portlets or pages, and specific authenticated visitors. This is a powerful mechanism for content personalization.

The entitlement segments are created in the E-Business Control Center, and are explained in detail in the *Guide to Using the E-Business Control Center* documentation. The process of associating entitlement segments with portal resources is explained in “Portal Administration Tools” in the *Getting Started with Portals and Portlets* documentation.

Rule-Based Entitlements Versus Rule-Based Personalization

WebLogic Portal enables rule-based personalization. Generally, with rule-based personalization, a rules engine is used to dynamically determine whether a user is part of a segment based on profile attributes, request or session attributes, or a time element. Based on this decision, specific content may be shown to the user. A developer uses JSP tags provided with WebLogic Portal, such as a placeholder or the content selector tag, to specify where on the site the personalized content should be displayed.

On the other hand, rule-based entitlements represent a specific usage of rule-based personalization at the system level. Entitlements are applied to specific portal resources: portlets and portal pages, rather than arbitrary content and areas on the site. The entitlements are controlled completely from administration tools, and no HTML/JSP developer involvement is required.

In addition, rule-based entitlements are typically used to control access to portal content, whereas rule-based personalization is used to serve targeted content. Inevitably, there will be scenarios where the line between rule-based entitlements and rule-based personalization will be blurred. For example, rule-based entitlements may be used to show a portal page with recommended content to a segment of users.

Both rule-based personalization and rule-based entitlements rely on the same set of infrastructure components—the BEA rules engine, the user profile, and property sets.

Index