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

Security Guide

BEA Campaign Manager for WebLogic 1.1
BEA WebLogic Commerce Server 3.5
BEA WebLogic Personalization Server 3.5
Document Edition 3.5
June 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Security Guide

Document Edition	Date	Software Version
3.5	June 2001	BEA Campaign Manager for WebLogic 1.1 BEA WebLogic Commerce Server 3.5 BEA WebLogic Personalization Server 3.5

Contents

3. Application Services and Security

4. Sample Applications, Administration Tools, and Security

Index

About This Document

This document provides information about security in the BEA Campaign Manager for WebLogic™, WebLogic Commerce Server™, and WebLogic Personalization Server™ products.

This document covers the following topics:

- Chapter 1, "Introduction," introduces the topic of security by explaining its importance to e-commerce applications, provides information that may help you to determine your security requirements, and provides an overview of Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server product security.
- Chapter 2, "Deployment Descriptors and Security Roles," describes what a deployment descriptor is, and explains the general XML syntax of the security-related elements these files contain. It also contains reference information for the deployment descriptors in the three sample applications. Finally, this chapter includes information about security roles, users, groups, and mapping security roles to principals.
- Chapter 3, "Application Services and Security," provides reference information about each of the EJB JAR files' deployment descriptors for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server enterprise application, as they relate to security roles and method permissions.
- Chapter 4, "Sample Applications, Administration Tools, and Security," describes how security is implemented in the sample applications (that is, the WebLogic Commerce Server sample storefront application, the WebLogic Personalization Server example portal application, and the Administration Tools application).

What You Need to Know

This document is intended mainly for individuals in the following roles:

- *Application Assembler*: The Application Assembler takes a set of components and assembles them into a complete J2EE application delivered in the form of a Enterprise ARchive (.EAR) file. The Application Assembler provides instructions that describe external dependencies of the application, which the Deployer will resolve in the deployment process.
- *Deployer*: The Deployer, usually expert in a specific operational environment, is responsible for deploying Web applications and Enterprise JavaBeans components into that environment. The deployment process is typically a three-stage process:
  - *Installation*: Moves the media to the server, generates the additional container-specific classes and interfaces that enable the container to manage the application components at runtime, and installs the application components and additional classes and interfaces into the J2EE containers.
  - *Configuration*: Resolves all the external dependencies and follows the application assembly instructions defined by the Application Assembler. For example, the Deployer is responsible for mapping the security roles defined by the Application Assembler to the user groups and accounts that exist in the operational environment into which the application components are deployed.
  - *Execution*: Starts up the newly installed and configured application.

  The Deployer's output is Web applications, Enterprise JavaBeans, applets, and application clients that have been customized for the target operational environment and are deployed in a specific J2EE container.
- *Site Administrator*: The Site Administrator is responsible for the configuration and administration of the enterprise's computing and networking infrastructure. The Site Administrator is also responsible for overseeing the run time well-being of the deployed J2EE applications.

Note: The roles described above are from the *Java 2 Platform Enterprise Edition Specification v1.3*, which you may consult for more information. For a quick reference, see the "Development Roles" section of the *J2EE Tutorial*.

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

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

Related Information

The following documents contain additional information about BEA WebLogic Server, Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server product security or supplemental information that may be helpful in understanding the contents of this *Security Guide*:

- *Deployment Guide*
- *Programming WebLogic Security*

- "web.xml Deployment Descriptor Elements" in the *Developing WebLogic Server Applications* documentation.

For more information about J2EE security, consult the following:

- *Java Servlet Specification v2.2*
- *Java 2 Platform Enterprise Edition Specification, v1.3*
- *Enterprise JavaBeans 1.1 Specification*

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 3.5 release.

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

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

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using

- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and 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`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introduction

The BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server are applications that may store personal information about customers or display Web site content based on a customer's identity. However, the level of sensitivity for such data varies. For example, a customer's credit card information is highly sensitive data that must be protected; a customer's color preferences may be considered by some as less sensitive.

Regardless of the perceived sensitivity of these activities, the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications provide you with ways to protect the confidentiality and integrity of customer data, customer preferences, and the overall integrity of customer transactions. This *Security Guide* was designed to help you understand how the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server leverage the security features of the Java™ 2 Platform Enterprise Edition (J2EE) specification and the J2EE-compliant security features of the WebLogic Server platform, as well as understand any additional security measures that have been established for application components. This guide also describes ways that you can modify security settings within the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server to inspire customer confidendence and solidify your e-business' online relationships.

As the introduction to this *Security Guide*, this topic includes the following sections:

- Determining Your Application Security Needs
- Development Roles
- Important Security Features
- Next Steps

Determining Your Application Security Needs

Security is a critical component of developing e-commerce applications that no organization can afford to ignore. Malicious users 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 abilitiy 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 user 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 discussion suggests, determining your application security needs can be a difficult balancing act. BEA recommends that you ask yourself the following questions when attempting to define your application security requirements:

- **What resources should I protect?**

  There are many resources in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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 users. Once a user decides to purchase an item, however, you may need to identify that user and retrieve some of their personal information. Obviously, this information must be protected from all other users of the Web site, who may or may not have malicious intent.

  Similarly, the tools included with the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications may allow 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 security ramifications of each resource will help you to properly protect it.

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

Development Roles

This *Security Guide* is intended mainly for individuals in the following roles:

- *Application Assembler*: The Application Assembler takes a set of components and assembles them into a complete J2EE application delivered in the form of a Enterprise ARchive (.EAR) file. The Application Assembler provides instructions that describe external dependencies of the application, which the Deployer will resolve in the deployment process.
- *Deployer*: The Deployer, usually expert in a specific operational environment, is responsible for deploying Web applications and Enterprise JavaBeans components into that environment. The deployment process is typically a three-stage process:
  - *Installation*: Moves the media to the server, generates the additional container-specific classes and interfaces that enable the container to manage the application components at runtime, and installs the application components and additional classes and interfaces into the J2EE containers.
  - *Configuration*: Resolves all the external dependencies and follows the application assembly instructions defined by the Application Assembler. For example, the Deployer is responsible for mapping the security roles defined

by the Application Assembler to the user groups and accounts that exist in the operational environment into which the application components are deployed.

- *Execution*: Starts up the newly installed and configured application.

The Deployer's output is Web applications, Enterprise JavaBeans, applets, and application clients that have been customized for the target operational environment and are deployed in a specific J2EE container.

- *Site Administrator*: The Site Administrator is responsible for the configuration and administration of the enterprise's computing and networking infrastructure. The Site Administrator is also responsible for overseeing the run time well-being of the deployed J2EE applications.

Note: The roles described above are from the *Java 2 Platform Enterprise Edition Specification v 1.3*, which you may consult for more information. For a quick reference, see the "Development Roles" section of the *J2EE Tutorial*.

Important Security Features

The important aspects of the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server security mechanisms are described in the following paragraphs.

Reliance on J2EE Standards and Platform Security

Security implementations in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications rely upon the security definitions in the Java 2 Enterprise Edition (J2EE) specification and security measures implemented in the WebLogic Server platform.

This dependancy has two important implications. First, the applications you build using the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server products as a base will also leverage the security mechanisms implemented by the container, instead of requiring you to embed security code within your Enterprise JavaBeans (EJBs) routines or their clients. This

architectural structure not only reduces debugging/maintenance issues and the likelihood of security holes due to programming errors, but it also avoids the requirement of tying security roles to individual applications. Second, because of the reliance on J2EE and the J2EE-compliant WebLogic Server, you can be certain that your Web applications will conform to the latest security standards in the industry.

For more information about J2EE security, see the *Java 2 Platform Enterprise Edition Specification, v1.3*. For more information about how the BEA WebLogic Server implements J2EE security, see *Programming WebLogic Security*.

Declarative Security with Deployment Descriptors

Because of the reliance on J2EE security and the WebLogic Server implementation of these techniques, security in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications is declarative. In other words, protections on resources are defined in a central configuration document called a deployment descriptor, which is then used by the WebLogic Server platform to enforce security restrictions. Access to protected resources is granted based on whether the requesting user's security role matches the role declared in the deployment descriptor for that resource.

Note: For more information about security roles, see Chapter 2, "Security Roles." For more information about deployment descriptors, see Chapter 2, "Deployment Descriptors and Security Roles."

A declarative approach to security enforcement has three major advantages. First, this type of security is fine-grained, meaning that access can be restricted all the way down to a specific method on a JavaBean. Second, the administrator of the WebLogic Server (or other J2EE-compliant server) can customize the security attributes for a particular production environment at deployment time. Third, because deployment descriptor information is declarative, it can be changed without modifying any JavaBean source code. At run time, the WebLogic Server simply reads the deployment descriptor and acts upon the component accordingly.

Note: For more information about declarative security, see the *Enterprise JavaBeans 1.1 Specification*, or the *Java 2 Platform Enterprise Edition Specification, v1.3*.

Next Steps

As the topics in this introduction show, the *Security Guide* assumes a certain level of familiarity with standard J2EE security implementations and with the security mechanisms of the BEA WebLogic Server. 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.

For more information about J2EE security, consult the following documents:

- *Java Servlet Specification v2.2*
- *Java 2 Platform Enterprise Edition Specification, v1.3*
- *Enterprise JavaBeans 1.1 Specification*

For more information about BEA WebLogic Server security, consult the following documents:

- *Programming WebLogic Security*
- "web.xml Deployment Descriptor Elements" in the *Developing WebLogic Server Applications* documentation.

Lastly, the *Deployment Guide* (which is part of the same Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server documentation set) may also be helpful in understanding the contents of this *Security Guide*.

If you feel comfortable with J2EE and WebLogic Server security topics, proceed through this guide — either sequentially or selectively — to answer your questions about Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server security.

2 Deployment Descriptors and Security Roles

Security for the BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server products is considered declarative because resource protections are centrally defined in configuration files called deployment descriptors, instead of within individual application components. This topic provides you with some general information about deployment descriptors, explains the syntax for the security-related elements you will find within the deployment descriptor files, and describes the deployment descriptor settings for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server product applications. Because an understanding of security roles is required to understand deployment descriptors, this topic will also be addressed.

This topic includes the following sections:

- The wlcs Application's Deployment Descriptors
- The exampleportal Application's Deployment Descriptors
- The tools Application's Deployment Descriptors

- About EJB 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 the type of JavaBean (session or entity) and the classes used for the remote, home, and bean class. It also specifies the transactional attributes of every method in the JavaBean, which security roles can access each method, and whether persistence in the entity beans is handled automatically or is preformed by the bean itself. For more detailed information about deployment descriptors, see the *Java 2 Platform Enterprise Edition Specification, v1.3*.

Note: For the purposes of this *Security Guide*, only the security-related aspects of a deployment descriptor will be discussed. For in-depth information about other elements in the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server deployment descriptors or for general information about deploying applications, see the *Deployment Guide*.

What Is a Security Role?

Because much of this document focuses on the deployment descriptor elements that define which security roles can access specific methods, you should know something about security roles. However, before you can understand security roles, you should know something about how the J2EE specification defines users and groups.

J2EE Users and Groups

A J2EE **user** is similar to an operating system user in that it represents a person. A **group** is a category of users, classified by common traits such as job title or customer profile. Categorizing users into groups makes it easier to control the access of large numbers of users. A J2EE group is scoped to the *entire* WebLogic Server.

Note: The Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server use the following WebLogic Server groups, which you can configure using the User Management Administration Tool:

- `wlcs_customer`
- `everyone`
- `admin`

For more information about the User Management Administration Tool, see "Using the User Management Tool" in the Creating and Managing Users chapter of the *Building Personalized Applications* documentation.

Roles and Principals

A security **role** is a named grouping of information resource access permissions, defined for an application. Although groups and roles both represent categories of users, a J2EE security role has a different scope that a group. Specifically, a security role is scoped only to a *specific application* in the WebLogic Server.

Note: For more infomation about assigning security roles in WebLogic Web applications, see "Deploying and Configuring Web Applications."

The J2EE security roles defined for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications are:

- `CustomerRole`—designated for customers who are registered with the e-business Web site.
- `AnonymousRole`—designated for anonymous access.
- `AdministrativeRole`—designated for administrators of the e-business application.

Mapping a **principal** to a 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 guest, which provides read-only access to a small subset of all of the application's resources. Any principal in the guest role would then have read-only access to only those few resources.

The J2EE security roles defined for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications map to the following WebLogic Server groups (principals), which may be defined in an application's `weblogic.xml` file:

- `CustomerRole` → `wlcs_customer` group
- `AnonymousRole` → `everyone` group
- `AdministrativeRole` → `admin` group

With this mapping, the `AdministrativeRole` always includes the `admin` group. Therefore, although you cannot modify access control lists (ACLs) at run time, you can provide specific users with access at run time simply by adding them to the appropriate group. For example, if you want to give user `JohnDoe` access to the JSP Administration Tools (which require administrative privileges), simply add `JohnDoe` to the `admin` group.

Notes: For more information about role to principal mapping in the weblogic.xml file, see "The weblogic.xml Deployment Descriptors" on page 2-12.

For more information about access control lists (ACLs), see "ACLs and Permissions" in the Security Fundamentals chapter of the *Programming WebLogic Security* documentation.

Deployment Descriptor Files in Enterprise Applications

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

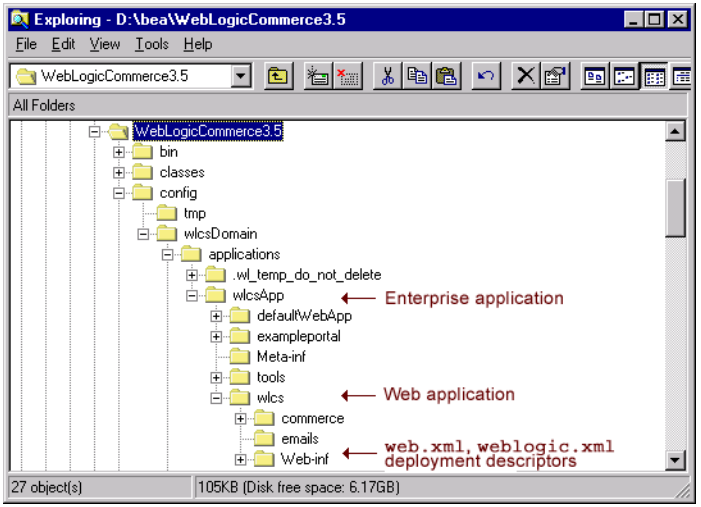
- One or more Web applications, which may or may not be saved as compressed Web ARchive (WAR) files,
- One or more Enterprise JavaBeans (EJBs), saved as compressed Java ARchive (JAR) files,
- And other resources.

Thus, although the deployment descriptor provides a central location for security-related deployment information, it is likely that an enterprise application will require more than one deployment descriptor file to communicate its security requirements to the server. In fact, for each individual Web application that comprises the enterprise application, there are two deployment descriptors: `web.xml` and `weblogic.xml`. For each EJB JAR that comprises the enterprise application, there are also two deployment descriptors: `ejb-jar.xml` and `weblogic-ejb-jar.xml`. Each of these deployment descriptor files are discussed in later sections of this chapter.

Location of Deployment Descriptor Files in the Product Directory Structure

The Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server product is essentially a collection of several prewritten Web applications and Enterprise JavaBeans (EJBs), organized into a single enterprise application. This enterprise application is called `wlcsApp`, and is located in the `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/applications` directory, where `$WL_COMMERCE_HOME` is the directory where you installed the product. The individual applications that comprise `wlcsApp` include `exampleportal`, `tools`, and `wlcs`, which are located in their own subdirectories under the `wlcsApp` directory. This directory structure, along with some additional subdirectories and files, is shown in Figure 2-1.

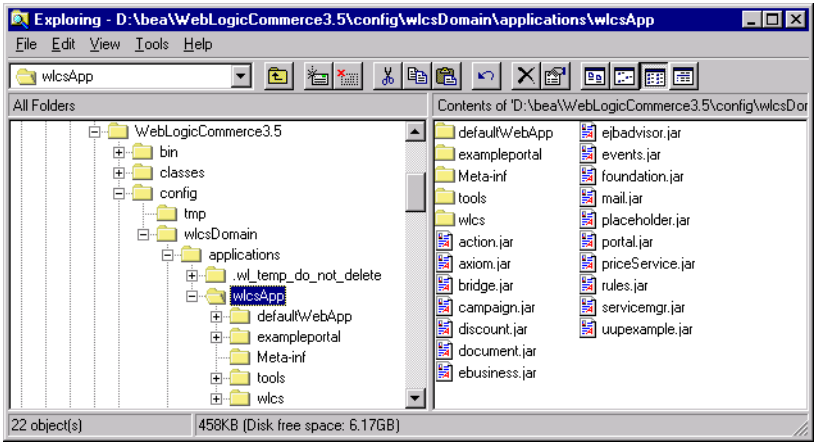
Figure 2-1 Location of Web Application Deployment Descriptors in the WebLogicCommerce3.5 Directory Structure



As Figure 2-1 also shows, in each application subdirectory (such as `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/applications/wlcsApp/wlcs`) there is a `WEB-INF` subdirectory. The `WEB-INF` directory contains the two deployment descriptors for the Web application: `web.xml` and `weblogic.xml`. The contents of the `web.xml` and `weblogic.xml` files for each of the prewritten applications are described later sections of this chapter.

As Figure 2-2 shows, the `wlcsApp` directory contains JAR files for all the Enterprise JavaBeans (EJBs) that make up the enterprise application. It is within each JAR file that the deployment descriptors `ejb-jar.xml` and `weblogic-ejb-jar.xml` can be found. The contents of the `ejb-jar.xml` and `weblogic-ejb-jar.xml` files for each of the prewritten EJBs are described in Chapter 3, "Application Services and Security."

Figure 2-2 Location of EJB Deployment Descriptors in the WebLogicCommerce3.5 Directory Structure



About the web.xml and weblogic.xml Application Deployment Descriptors

As previously described, each Web application has two deployment descriptors: `web.xml` and `weblogic.xml`. This section provides a general description of these files, then lists the contents of these files (that are relevant to security) for the `wlcs`, `exampleportal`, and `tools` applications.

The web.xml Deployment Descriptors

Along with other information, the `web.xml` application deployment descriptor contains several sets of elements for implementing the J2EE declarative security model. These deployment descriptors are located in each application's `WEB-INF` directory. Each `web.xml` deployment descriptor may contain elements for each of the following security-related topics:

- Port Numbers and Security Constraints for Generated URLs
  - Generate Port Numbers for
  - Determine Which Links Use HTTPS
- Session Timeout
- Declarations of Secure JSPs

These topics are described in detail in the following sections.

Note: For more information about the XML elements this section describes, refer to "web.xml Deployment Descriptor Elements" in the *Developing WebLogic Server Applications* documentation.

Port Numbers and Security Constraints for Generated URLs

This section describes the elements in the `web.xml` deployment descriptor that configure the `createWebflowURL()` method to do the following:

- Generate Port Numbers for
- Determine Which Links Use HTTPS

Note: For more information about the `createWebflowURL()` method, see "Using Webflow in Your Web Pages," which is located in the "Customizing Webflow and Pipelines" chapter of the *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline* documentation.

Generate Port Numbers for

HTTP and HTTPS

The `<context-param>` element syntax shown in Listing 2-1 determine which port numbers the `createWebflowURL()` method encodes in the URLs that it generates.

Listing 2-1 Elements That Generate Port Numbers

```
<context-param>
 <param-name>HTTP_PORT</param-name>
 <param-value>port-number</param-value>
</context-param>
```

```
<context-param>
   <param-name>HTTPS_PORT</param-name>
   <param-value>port-number</param-value>
</context-param>
```

Determine Which Links Use HTTPS

The `<context-param>` element syntax shown in Listing 2-2 declares a set of files, pipelines, and input processors that need to be accessed via HTTPS.

Listing 2-2 Elements That Determine Which Links Use HTTPS

```
<context-param>
   <param-name>HTTPS_URL_PATTERNS</param-name>

   <param-value>
      URLs-and-URL patterns
   </param-value>

</context-param>
```

When the `createWebflowURL()` method encounters one of the resources you declare in the `<param-value>` subelement, it generates a URL that uses the HTTPS protocol.

Session Timeout

The `<session-config>` element syntax shown in Listing 2-3 specifies how many minutes of inactivity the server will tolerate before ending the session.

Listing 2-3 Element That Specifies Session Timeout

```
<session-config>

   <session-timeout>number of minutes until
    timeout</session-timeout>

</session-config>
```

Declarations of Secure JSPs

The <security-constraint> element syntax shown in Listing 2-4 declares a collection of resources (JSPs) that only specific security roles can access.

Listing 2-4 Elements That Declare Secure JSPs

```
<security-constraint>

   <web-resource-collection>

       <web-resource-name>name of the resource collection
       </web-resource-name>

       <description>documentation of the collection's scope and
       purpose</description>

       <url-pattern>single URL pattern: use as many url-pattern
       elements as you need. One pattern/filename per element.
       </url-pattern>

       <http-method>GET</http-method>
       <http-method>POST</http-method>

   </web-resource-collection>


   <auth-constraint>

       <description>documentation of the authorization
       constraint</description>

       <role-name>name of declared security role: use as many
       role-name elements as you need. One role per element.
       </role-name>

   </auth-constraint>


   <user-data-constraint>

       <transport-guarantee>INTEGRAL or CONFIDENTIAL;
       see below for details</transport-guarantee>

   </user-data-constraint>

</security-constraint>
```

Use one of the following values in the <transport-guarantee> element:

- INTEGRAL to prevent the data from being changed while transmitting between the client and server.
- CONFIDENTIAL to prevent other entities from observing the contents of the transmission.

Additionally, any roles used within the <auth-contraint> portion of the element must also be declared within the web.xml deployment descriptor, using the following syntax shown in Listing 2-5.

Listing 2-5 Elements That Declare Security Roles

```
<security-role>

   <description>declaration documentation for the security roles
   used in the <auth-contraint> element</description>

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

</security-role>
```

To give another role access to the resource collection, add <role-name> elements to the <auth-constraint> element, and add the role to the <security-role> declaration.

You can also add <url-pattern> elements to specify new directories or to specify specific files. Note that a pattern or filename must start with a / (forward slash) character. For more information on specifying URL patterns, refer to the *Java Servlet Specification v2.2*.

Note: If you add or modify any <url-pattern> elements, and if the transport-guarantee is set to INTEGRAL/CONFIDENTIAL, be sure to update those patterns in the <context-param> element for SSL.

The weblogic.xml Deployment Descriptors

Along with the `web.xml` deployment descriptor, each J2EE component application also requires a `weblogic.xml` deployment descriptor, which declares deployment properties specific to the WebLogic Server. The `weblogic.xml` deployment descriptor is located in the same directory as `web.xml` (that is, in the application's `WEB-INF` subdirectory).

Among other elements, the `weblogic.xml` file contains the element syntax shown in Listing 2-6, which maps a J2EE security role to a user or group (principal) that a security realm uses to define access control lists (ACLs).

Listing 2-6 Elements That Map Roles to Principals

```
<security-role-assignment>

   <role-name>J2EE role</role-name>
   <principal-name>security-realm user or group</principal-name>

</security-role-assignment>
```

Notes: For more information about security realms and ACLs in the WebLogic Server, see "Security Fundamentals" in the *Programming WebLogic Security* documentation.

For more information about the non-security related elements in weblogic.xml, see "The weblogic.xml File," located in "The Reference Domain" chapter of the *Deployment Guide*.

The wlcs Application's Deployment Descriptors

As described in "The web.xml Deployment Descriptors" on page 2-7, the e-commerce (`wlcs`) application's `web.xml` deployment descriptor specifies the following security-related information:

- Port Numbers for HTTP and HTTPS

- Links Using HTTPS
- Session Timeout
- Secure JavaServer Pages (JSPs)

As described in "The weblogic.xml Deployment Descriptors" on page 2-12, the `wlcs` application's `weblogic.xml` deployment descriptor specifies the following:

- Role to Principal Mapping

Port Numbers for HTTP and HTTPS

The default port numbers for HTTP and HTTPS defined in the `wlcs` Web application's `web.xml` deployment descriptor are `7501` and `7502`, respectively.

If you want the `wlcs` Web application to use the same port numbers as the WebLogic Server instance (for example, if you are using WebLogic Server as your HTTP server), either deactivate these elements by surrounding them in comment tags (`<!-- -->`), or change the values to match the listen port properties for the server. If you want the `wlcs` Web application to use different port numbers (for example, if you want to use the port numbers of a proxy server), change the values shown in the appropriate element. For more information, refer to "Changing the Listen Ports" in the *Deployment Guide*.

Links Using HTTPS

The links in the `wlcs` Web application that are defined in the `web.xml` deployment descriptor to use HTTPS are shown in Listing 2-7.

Listing 2-7 wlcs Application Links That Use HTTPS

```
<context-param>
   <param-name>HTTPS_URL_PATTERNS</param-name>

   <param-value>
     /commerce/user/*,/commerce/order/*,/commerce/register/*,
     newuser_*,profilenewcc_*,paymentnewcc_*,profileeditcc_*,
     paymenteditcc_*,viewprofile_*,editprofile_*,
     editdemographics_*,profilenewaddress_*,profileeditaddress_*,
     changepassword_*,EnterShippingInfo_*,SelectShippingAddress_*,
     AddNewShippingAddress_*,orderhistory_*,RefreshOrderHistory_*,
```

```
      AuthorizePayment,CommitOrder,RefreshPaymentHistory,
      DeleteCreditCard,TaxVerifyShippingAddress_*,
      shoppingcart_InitShippingMethodList
    </param_value>

</context-param>
```

If desired, you can add specific resource names or name patterns to the `<param-value>` element. Specify patterns for Pipelines and input processors in the form of *`pattern`*`_*`. For example, to enable SSL for all requests to input processors whose names start with `profileeditcc_`, use the pattern `profileeditcc_*`.

Notes: When naming specific Pipelines and input processors, do not use the `.inputprocessor` or `.pipeline` extension

If you add any target names or patterns to the `<param-value>` element, you must also add them to the `<security-constraint>` element, which is described in "Declarations of Secure JSPs" on page 2-10.

Session Timeout

By default, the `$WL_COMMERCE_HOME/server/webapps/wlcs/WEB-INF/web.xml` deployment descriptor sets the parameter to 15 minutes. You can modify this setting depending on your security needs.

Secure JavaServer Pages (JSPs)

As shown in Listing 2-8, the `wlcs web.xml` deployment descriptor declares a single role, `CustomerRole`, in the `<role-name>` element, and several `<url-pattern>` elements.

Listing 2-8 Secure JSPs for the wlcs Application

```
<!-- Configuration for customer self administration pages
     Configuration for role: Customer Role -->

<security-constraint>

   <web-resource-collection>
```

```
        <!-- Define a resource collection -->

        <web-resource-name>Customer Profile - Self Administration
        Pages</web-resource-name>
        <description>Customer Profile - Self Administration Pages
        </description>

        <!-- URL pattern for the resource collection -->

        <url-pattern>/commerce/user/*</url-pattern>
        <url-pattern>/commerce/order/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>

    </web-resource-collection>


    <!-- This contraint applies to users with role “CustomerRole” -->

    <auth-constraint>
        <description>Users with role “CustomerRole”</description>
        <role-name>CustomerRole</role-name>
    <auth-contraint>


    <!-- For enabling SSL, specify CONFIDENTIAL or INTEGRAL -->

    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

</security-constraint>


<!-- Configuration for customer self registration pages -->

<security-constraint>

    <web-resource-collection>

        <!-- Define a resource collection -->

        <web-resource-name>Customer Self Registration
        </web-resource-name>
        <description>Customer Self Registration Pages
        </description>

        <!-- URL pattern for the resource collection -->
```

```
        <url-pattern>/commerce/register/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>

    </web-resource-collection>


    <!-- For enabling SSL, specify CONFIDENTIAL or INTEGRAL -->

    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

</security-constraint>

.
.
.

<!-- Declare all the roles used in the <auth-constraint> element
above -->

<security-role>
   <description>Registered customers with role CustomerRole
   </description>
   <role-name>CustomerRole</role-name>
</security-role>
```

Role to Principal Mapping

The e-commerce (`wlcs`) application's `weblogic.xml` deployment descriptor specifies the security-related information shown in Listing 2-9.

Listing 2-9 wlcs Security Role to Principal Mapping

```
<!-- Role mapping to the realm groups: The following elements
 declare mappings between J2EE roles and WLCS groups -->

<!-- J2EE role CustomerRole maps to WLCS group "wlcs_customer" -->

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

The exampleportal Application's Deployment Descriptors

As described in "The web.xml Deployment Descriptors" on page 2-7, the `exampleportal` (ACME) application's `web.xml` deployment descriptor specifies the following security-related information:

- Secure JavaServer Pages (JSPs)

As described in "The weblogic.xml Deployment Descriptors" on page 2-12, the `exampleportal` application's `weblogic.xml` deployment descriptor specifies the following:

- Role to Principal Mapping

Secure JavaServer Pages (JSPs)

As shown in Listing 2-10, the `exampleportal` `web.xml` deployment descriptor declares a single role, `PortalUserRole`, in the `<role-name>` element, and one `<url-pattern>` element.

Listing 2-10 Secure JSPs for the exampleportal Application

```
<security-constraint>

   <web-resource-collection>

      <web-resource-name>Portal Resources</web-resource-name>

      <description>The Portal user logged-in Pages. Note that since
       all portal pages (both in a logged-in state and a
       non-logged-in state) are accessed via the same URL
       (portal.jsp), you cannot uniquely trigger authorization on
       that URL. Instead,the _userlogin.jsp resource is used.
      </description>

      <url-pattern>/portals/repository/_userlogin.jsp
      </url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>

   </web-resource-collection>
```

```
   <auth-constraint>
      <description>Portal Users</description>
      <role-name>PortalUserRole</role-name>
   </auth-constraint>

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

</security-constraint>

.
.
.
<security-role>
   <description>Portal Users</description>
   <role-name>PortalUserRole</role-name>
</security-role>
```

Role to Principal Mapping

The `exampleportal` (Acme) application's `weblogic.xml` deployment descriptor specifies the security-related information shown in Listing 2-11.

Listing 2-11 exampleportal Security Role to Principal Mapping

```
<!-- The Portal User Role mapping to the Security Realm -->

<security-role-assignment>
   <role-name>PortalUserRole</role-name>
   <principal-name>AcmeUsers</principal-name>
   <principal-name>wlcs_customer</principal-name>
</security-role-assignment>
```

The tools Application's Deployment Descriptors

As described in "The web.xml Deployment Descriptors" on page 2-7, the `tools` application's `web.xml` deployment descriptor specifies the following security-related information:

- Secure JavaServer Pages (JSPs)

As described in "The weblogic.xml Deployment Descriptors" on page 2-12, the `tools` application's `weblogic.xml` deployment descriptor specifies the following:

- Role to Principal Mapping

Secure JavaServer Pages (JSPs)

As shown in Listing 2-12, the `tools` `web.xml` deployment descriptor declares a single role, `AdminRole`, in the `<role-name>` element, and three `<url-pattern>` elements.

Listing 2-12 Secure JSPs for the tools Application

```
<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>AdminRole</role-name>
   </auth-constraint>

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

</security-constraint>

.
.
.
```

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

Role to Principal Mapping

The `tools` application's `weblogic.xml` deployment descriptor specifies the security-related information shown in Listing 2-13.

Listing 2-13 tools Security Role to Principal Mapping

```
<!-- The Admin Role mapping to the WebLogic Realm -->

<security-role-assignment>
   <role-name>AdminRole</role-name>
   <principal-name>admin</principal-name>
</security-role-assignment>
```

About EJB Deployment Descriptors

Each JAR scoped to an enterprise application contains two associated XML deployment descriptors: `ejb-jar.xml` and `weblogic-ejb-jar.xml`. It is in these XML files that an application's individual JavaBeans are registered with appropriate security constraints.

Note: Because the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server enterprise application consists of many EJB JAR files, the detailed contents of the EJB deployment descriptors can be found in Chapter 3, "Application Services and Security."

The ejb-jar.xml Deployment Descriptors

An `ejb-jar.xml` file is the primary deployment descriptor for an Enterprise JavaBean (EJB). 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 2-14.

Listing 2-14 ejb-jar.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 2-14 also shows, the <enterprise-beans> element contains <entity> subelements when declaring entity beans, and/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.

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 corrresponding 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. However, 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.

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 2-15 shows the syntax of the security-role assignment element.

Listing 2-15 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 "About the web.xml and weblogic.xml Application Deployment Descriptors" on page 2-7.

For more information about security roles and principals, see "Roles and Principals" on page 2-3.

3 Application Services and Security

As described in Chapter 2, "Deployment Descriptors and Security Roles," 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server application services as a starting point for developing your own applications, this topic describes the contents of these deployment descriptors for each JAR in the included `wlcsApp` enterprise application. Therefore, this topic includes the following sections:

- action.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- axiom.jar
  - Enterprise Bean Definitions
  - Security-Role References
  - Assembly Descriptor
  - Security-Role Assignments
- bridge.jar
  - Enterprise Bean Definitions

- Assembly Descriptor
- Security-Role Assignments

- campaign.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- discount.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- document.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- ebusiness.jar
  - Enterprise Bean Definitions
  - Security-Role References
  - Assembly Descriptor
  - Security-Role Assignments
- ejbadvisor.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- events.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments

- foundation.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- mail.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- placeholder.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- portal.jar
  - Enterprise Bean Definitions
  - Security-Role References
  - Assembly Descriptor
  - Security-Role Assignments
- priceService.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- rules.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments
- servicemgr.jar
  - Enterprise Bean Definitions

  - Assembly Descriptor
  - Security-Role Assignments
- uupexample.jar
  - Enterprise Bean Definitions
  - Assembly Descriptor
  - Security-Role Assignments

Notes: The JAR files containing the `ejb-jar.xml` and `weblogic-ejb-jar.xml` deployment descriptors that are described in this chapter can be found in the `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/wlcsApp` subdirectory, where `$WL_COMMERCE_HOME` is the directory where you installed the product.

Only the security-related elements of the `ejb-jar.xml` and `weblogic-ejb-jar.xml` deployment descriptors are discussed in this chapter. For more detailed information about the other elements these deployment descriptors contain, see the *Deployment Guide*.

action.jar

The `action.jar` file contains an EJB that provides the executable actions for the campaign server (that is, the mail action, the ads action, and the offer discount action).

Enterprise Bean Definitions

Table 3-1 lists the enterprise JavaBean that is defined within the `action.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-1 action.jar ejb-jar.xml EJB Definition

Name	Type
`ActionService`	`Stateless session`

Assembly Descriptor

Within the `action.jar` file, the `ejb-jar.xml` deployment descriptor registers the `ActionService` stateless session JavaBean with the application assembly descriptor shown in Listing 3-1.

Listing 3-1 Assembly Descriptor Element for the ActionService JavaBean

```
<assembly descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

       <role-name>AnonymousRole</role-name>

       <method>
          <ejb-name>ActionService</ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name>
       </method>

   </method-permission>

...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all of the methods in the `ActionService` JavaBean's `Home` interface.

Security-Role Assignments

Within the `action.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `ActionService` stateless session JavaBean, as shown in Listing 3-2. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-2 Security Role Assignments for the ActionService JavaBean

```
<security-role-assignment>
   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>
</security-role-assignment>
```

axiom.jar

The `axiom.jar` file is a collection of EJBs that provide user and group management and Unified User Profile (UUP) services. For more information, see "Unified User Profiles" in the Creating and Managing Users chapter of the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-2 lists the enterprise JavaBeans that are defined within the `axiom.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-2 axiom.jar ejb-jar.xml EJB Definitions

Name	Type
com.beasys.commerce.axiom.contact.UserManager	Stateless session
com.beasys.commerce.axiom.contact.RealmConfiguration	Stateless session
com.beasys.commerce.axiom.contact. UnifiedProfileTypeManager	Stateless session
com.beasys.commerce.axiom.contact.User	Entity
com.beasys.commerce.axiom.contact.Group	Entity

Security-Role References

In addition, the `ejb-jar.xml` deployment descriptor contains security-role reference elements for the `com.beasys.commerce.axiom.contact.UserManager` stateless session JavaBean and the `com.beasys.commerce.axiom.contact.User` entity bean. These elements, shown in Listing 3-3, enable the EJB to do programmatic security checking, if such behavior is desired.

Listing 3-3 Security-Role References in the User and UserManager JavaBeans

```
<security-role-ref>
   <description>This ref declares the Administrative role for this
   bean</description>
   <role-name>AdministrativeRole</role-name>
   <role-link>AdministrativeRole</role-link>
</security-role-ref>
```

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

Assembly Descriptor

Within the `axiom.jar` file, the `ejb-jar.xml` deployment descriptor registers the Axiom stateless session and entity JavaBeans with the application assembly descriptor shown in Listing 3-4.

Listing 3-4 Assembly Descriptor Element for the Axiom JavaBeans

```
<assembly descriptor>

   <security-role>
      <description>Administrators</description>
      <role-name>AdministrativeRole</role-name>
   </security-role>

   <security-role>
      <description>Anonymous Users</description>
      <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

      <role-name>AnonymousRole</role-name>

      <method>
         <ejb-name>com.beasys.commerce.axiom.contact.
         UserManager</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.beasys.commerce.axiom.contact.
         UserManager</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.beasys.commerce.axiom.contact.
         RealmConfiguration</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>
```

```
        <method>
           <ejb-name>com.beasys.commerce.axiom.contact.
           RealmConfiguration</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>*</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.axiom.contact.
           UnifiedProfileTypeManager</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.axiom.contact.
           UnifiedProfileTypeManager</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>*</method-name>
        </method>

     </method-permission>

...

</assembly-descriptor>
```

In this case, the assembly descriptor first specifies two security roles, `AdministrativeRole` and `AnonymousRole`. The assembly descriptor then specifies the individual method permissions for the three stateless session JavaBeans in the `axiom.jar` file (`UserManager`, `RealmConfiguration`, and `UnifiedProfileTypeManager`) according to the `AnonymousRole`. Thus, users within the security role `AnonymousRole` will be granted access to all of the methods in the `UserManager`, `RealmConfiguration`, and `UnifiedProfileTypeManager` JavaBeans' `Home` interfaces, as well as all of the methods of their `Remote` interfaces. The `AdministrativeRole` is not used.

Security-Role Assignments

Within the `axiom.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the Axiom JavaBeans, as shown in Listing 3-5. In this case, the security role `AdministrativeRole` is assigned to the `admin` principal, and the `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-5 Security Role Assignments for the Axiom JavaBeans

```
<security-role-assignment>

  <role-name>AdministrativeRole</role-name>
  <principal-name>admin</principal-name>

</security-role-assignment>

<security-role-assignment>

  <role-name>AnonymousRole</role-name>
  <principal-name>everyone</principal-name>

</security-role-assignment>
```

bridge.jar

The `bridge.jar` file contains an EJB that provides the LDAP configuration information for the Unified User Profile (UUP) service. For more information about UUP, see "Unified User Profiles" in the Creating and Managing Users chapter of the *Building Personalized Applications* documentation. For more information about UUP and LDAP, see "Using the LDAP Realm" in the same chapter.

Enterprise Bean Definitions

Table 3-3 lists the enterprise JavaBean that is defined within the `bridge.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-3 bridge.jar ejb-jar.xml EJB Definition

Name	Type
`com.beasys.commerce.bridge.ldap.LDAPConfiguration`	`Stateless session`

Assembly Descriptor

Within the `bridge.jar` file, the `ejb-jar.xml` deployment descriptor registers the `LDAPConfiguration` stateless session JavaBean with the application assembly descriptor shown in Listing 3-6.

Listing 3-6 Assembly Descriptor Element for the LDAPConfiguration JavaBean

```
<assembly descriptor>

      <security-role>
         <description>Anonymous Users</description>
         <role-name>AnonymousRole</role-name>
      </security-role>

      <method-permission>

         <role-name>AnonymousRole</role-name>

         <method>
            <ejb-name>com.beasys.commerce.bridge.ldap.
            LDAPConfiguration</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
         </method>

      </method-permission>

   ...

</assembly descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all of the methods in the `LDAPConfiguration` JavaBean's `Home` interface.

Security-Role Assignments

Within the `bridge.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `LDAPConfiguration` stateless session JavaBean, as shown in Listing 3-7. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-7 Security Role Assignments for the LDAPConfiguration JavaBean

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

campaign.jar

The `campaign.jar` file is a collection of EJBs that provide the campaign and scenario services and repositories.

Enterprise Bean Definitions

Table 3-4 lists the enterprise JavaBeans that are defined within the `campaign.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-4 campaign.jar ejb-jar.xml EJB Definitions

Name	Type
`com.beasys.commerce.campaign.ScenarioService`	`Stateless session`
`com.beasys.commerce.campaign.ScenarioRepository`	`Stateless session`
`com.beasys.commerce.campaign.CampaignService`	`Stateless session`
`com.beasys.commerce.campaign.CampaignRepository`	`Stateless session`

Assembly Descriptor

Within the `campaign.jar` file, the `ejb-jar.xml` deployment descriptor registers the Campaign stateless session JavaBeans with the application assembly descriptor shown in Listing 3-8.

Listing 3-8 Assembly Descriptor Element for the Campaign JavaBeans

```
<assembly-descriptor>

      <security-role>
         <description>Anonymous Users</description>
         <role-name>AnonymousRole</role-name>
      </security-role>

      <security-role>
         <description>Administrative Users</description>
         <role-name>AdminRole</role-name>
      </security-role>


      <!-- Permissions for the AnonymousRole -->

      <method-permission>
```

```
        <role-name>AnonymousRole</role-name>

        <method>
           <ejb-name>com.bea.commerce.campaign.
           ScenarioService</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
        </method>

        <method>
           <ejb-name>com.bea.commerce.campaign.
           ScenarioService</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>*</method-name>
        </method>

       <method>
          <ejb-name>com.bea.commerce.campaign.
          CampaignService</ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name>
       </method>

       <method>
          <ejb-name>com.bea.commerce.campaign.
          CampaignService</ejb-name>
          <method-intf>Remote</method-intf>
          <method-name>*</method-name>
       </method>

       <method>
          <ejb-name>com.bea.commerce.campaign.
          CampaignRepository</ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name>
       </method>

       <method>
          <ejb-name>com.bea.commerce.campaign.
         CampaignRepository</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>getCampaignService</method-name>
       </method>

  </method-permission>


  <!-- Permissions for the AdminRole -->

  <method-permission>
```

```
<role-name>AdminRole</role-name>

<method>
   <ejb-name>com.bea.commerce.campaign.
   ScenarioService</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   ScenarioService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   ScenarioRepository</ejb-name>
  <method-intf>Home</method-intf>
  <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   ScenarioRepository</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   CampaignService</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   CampaignService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.campaign.
   CampaignRepository</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>
```

```
        <method>
           <ejb-name>com.bea.commerce.campaign.
           CampaignRepository</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>*</method-name>
        </method>

    </method-permission>

  ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `Home` and `Remote` interfaces for the `ScenarioService` and `CampaignService` JavaBeans, plus all methods on the `Home` interface of the `CampaignRepository` JavaBean. In addition, users in the `AnonymousRole` will be granted access to the `getCampaignService` method on the `Remote` interface of the `CampaignRepository` JavaBean. Users within the security role `AdminRole` will be granted access to all methods in the `Home` and `Remote` interfaces for all the JavaBeans.

Security-Role Assignments

Within the `campaign.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the Campaign stateless session JavaBeans, as shown in Listing 3-9. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal, and the `AdminRole` is assigned to both the `admin` and `system` principals.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-9 Security Role Assignments for the Campaign JavaBeans

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>
```

```
</security-role-assignment>


<security-role-assignment>

    <role-name>AdminRole</role-name>
    <principal-name>admin</principal-name>
    <principal-name>system</principal-name>

</security-role-assignment>
```

discount.jar

The `discount.jar` file is a collection of EJBs that provide the discount definition and association services. More information about the Discount Management and Association services can be found in the "Discounts" chapter of the *Managing Purchases and Processing Orders* documentation.

Enterprise Bean Definitions

Table 3-5 lists the enterprise JavaBeans that are defined within the `discount.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-5 discount.jar ejb-jar.xml EJB Definitions

Name	Type
`DiscountManagement`	`Stateless session`
`DiscountAssociationMgr`	`Stateless session`
`DiscountSet`	`Entity`
`Discount`	`Entity`
`DiscountAssociation`	`Entity`

Assembly Descriptor

Within the `discount.jar` file, the `ejb-jar.xml` deployment descriptor registers the Discount stateless session and entity JavaBeans with the application assembly descriptor shown in Listing 3-10.

Listing 3-10 Assembly Descriptor Element for the Discount JavaBeans

```
<assembly-descriptor>

      <security-role>
         <description>Anonymous Users</description>
         <role-name>AnonymousRole</role-name>

      </security-role>

      <method-permission>

         <role-name>AnonymousRole</role-name>

         <method>
           <ejb-name>DiscountManagement</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
         </method>

         <method>
           <ejb-name>DiscountManagement</ejb-name>
           <method-name>*</method-name>
         </method>

         <method>
            <ejb-name>DiscountAssociationMgr</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
         </method>

         <method>
            <ejb-name>DiscountAssociationMgr</ejb-name>
            <method-name>*</method-name>
         </method>

      </method-permission>

   ...

</assembly-descriptor>
```

Note: The <method> tag without a <method-intf> specifies the <method-name> on both the Home and Remote interfaces.

In this case, the assembly descriptor specifies that users within the security role AnonymousRole will be granted access to all the methods in the Home and Remote interfaces for the DiscountManagement and DiscountAssociationMgr JavaBeans.

Security-Role Assignments

Within the discount.jar file, the weblogic-ejb-jar.xml deployment descriptor defines the security role assignments for the Discount stateless session and entity JavaBeans, as shown in Listing 3-11. In this case, the security role AnonymousRole is assigned to the everyone principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-11 Security Role Assignments for the Discount JavaBeans

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

document.jar

The document.jar file is a collection of EJBs that provide the document management services (searching, retrieval, and schemas).

Enterprise Bean Definitions

Table 3-6 lists the enterprise JavaBeans that are defined within the `document.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-6 document.jar ejb-jar.xml EJB Definitions

Name	Type
`com.beasys.commerce.axiom.document.Document`	`Entity`
`com.beasys.commerce.axiom.document.DocumentSchema`	`Entity`
`com.beasys.commerce.axiom.document.DocumentManager`	`Stateless session`

Assembly Descriptor

Within the `document.jar` file, the `ejb-jar.xml` deployment descriptor registers the Document stateless session and entity JavaBeans with the application assembly descriptor shown in Listing 3-12.

Listing 3-12 Assembly Descriptor Element for the Document JavaBeans

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

       <role-name>AnonymousRole</role-name>

       <method>
          <ejb-name>com.beasys.commerce.axiom.document.Document
          </ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name>
       </method>
```

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

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

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

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

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

    </method-permission>

  ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `Home` and `Remote` interfaces for each of the Document JavaBeans.

Security-Role Assignments

Within the `document.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the Document stateless session and entity JavaBeans, as shown in Listing 3-13. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-13 Security Role Assignments for the Document JavaBeans

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

ebusiness.jar

The `ebusiness.jar` file is a collection of EJBs that provide the commerce services, including product catalog, order, tax calculation, shipping, payment, and supporting EJB Pipeline components. For more information about these services, see the *Managing Purchases and Processing Orders* documentation.

Enterprise Bean Definitions

Table 3-7 lists the enterprise JavaBeans that are defined within the `ebusiness.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-7 ebusiness.jar ejb-jar.xml EJB Definitions

Name	Type
com.beasys.commerce.ebusiness.security.Encryptor	Stateless session
com.beasys.commerce.ebusiness.security.Decryptor	Stateless session
com.beasys.commerce.ebusiness.payment.CreditCardService	Stateless session
com.beasys.commerce.ebusiness.payment.PaymentTransaction	Entity
com.beasys.commerce.ebusiness.tax.taxware.TaxwareTaxCalculator	Stateful session
com.beasys.commerce.ebusiness.customer.Customer	Entity
com.beasys.commerce.ebusiness.catalog.CatalogManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.category.CategoryManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.query.CatalogQueryManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.category.JdbcCategoryManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.data.EpmCustomDataManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.item.JdbcProductItemManager	Stateless session
com.beasys.commerce.ebusiness.catalog.service.query.JdbcCatalogQueryManager	Stateless session
com.beasys.commerce.ebusiness.shoppingcart.pipeline.DeleteProductItemFromSavedListPC	Stateless session

Table 3-7 ebusiness.jar ejb-jar.xml EJB Definitions (Continued)

Name	Type
com.beasys.commerce.ebusiness.shoppingcart.pipeline.RefreshSavedListPC	Stateless session
com.beasys.commerce.ebusiness.shoppingcart.pipeline.MoveProductItemToSavedListPC	Stateless session
com.beasys.commerce.ebusiness.shoppingcart.pipeline.MoveProductItemToShoppingCartPC	Stateless session
com.beasys.commerce.ebusiness.shipping.ShippingMethod	Entity
com.beasys.commerce.ebusiness.shipping.ShippingHelper	Stateless session
com.beasys.commerce.ebusiness.order.Order	Entity
com.beasys.commerce.ebusiness.order.OrderManager	Stateless session

Security-Role References

In addition, the `ejb-jar.xml` deployment descriptor contains security-role reference elements for the E-business stateless session and entity JavaBeans. These elements, shown in Listing 3-14 through Listing 3-17, enable the EJBs to do programmatic security checking, if such behavior is desired.

Listing 3-14 illustrates only the Anonymous security-role reference, as it is defined for the following E-business JavaBeans:

- DeleteProductItemFromSavedListPC
- Encryptor
- MoveProductItemToSavedListPC
- MoveProductItemToShoppingCartPC
- RefreshSavedListPC
- ShippingHelper
- ShippingMethod

- TaxwareTaxCalculator

Listing 3-14 Anonymous Security-Role Reference in the E-Business JavaBeans

```
<security-role-ref>

   <description>This ref declares the Anonymous role for this
    bean</description>

   <role-name>AnonymousRole</role-name>
   <role-link>AnonymousRole</role-link>

</security-role-ref>
```

Note: In the `TaxwareTaxCalculator` JavaBean's security-role reference, the `<description>` subelement incorrectly reads `AdministrativeRole` instead of `AnonymousRole`. However, because the description is essentially a comment, the server is not affected by it.

Listing 3-15 illustrates the Customer and Administrative security-role references, as they are defined for the following E-business JavaBeans:

- CreditCardService
- Decryptor
- Order
- OrderManager
- PaymentTransaction

Listing 3-15 Customer and Administrative Security-Role References in the E-Business JavaBeans

```
<security-role-ref>

   <description>This ref declares the Customer role for this
    bean</description>

   <role-name>CustomerRole</role-name>
   <role-link>CustomerRole</role-link>
```

```
</security-role-ref>

<security-role-ref>

  <description>This ref declares the Administrative role for this
   bean</description>

  <role-name>AdministrativeRole</role-name>
  <role-link>AdministrativeRole</role-link>

</security-role-ref>
```

Listing 3-16 Anonymous and Administrative Security-Role References in the Customer JavaBean

```
<security-role-ref>

   <description>This ref declares the Anonymous role for this
    bean</description>

   <role-name>AnonymousRole</role-name>
   <role-link>AnonymousRole</role-link>

</security-role-ref>

<security-role-ref>

   <description>This ref declares the Administrative role for
    this bean</description>

   <role-name>AdministrativeRole</role-name>
   <role-link>AdministrativeRole</role-link>

</security-role-ref>
```

Listing 3-17 illustrates the Administrative, Customer, and Anonymous security-role references, as they are defined for the following E-business JavaBeans:

- CatalogManager
- CategoryManager
- JdbcCategoryManager
- JdbcProductItemManager

- ProductItemManager

Listing 3-17 Administrative, Customer, and Anonymous Security-Role References in the E-Business JavaBeans

```
<security-role-ref>

   <description>This ref declares the Administrative role for this
    bean</description>

   <role-name>AdministrativeRole</role-name>
   <role-link>AdministrativeRole</role-link>

</security-role-ref>

<security-role-ref>

   <description>This ref declares the Customer role for this
    bean</description>

   <role-name>CustomerRole</role-name>
   <role-link>CustomerRole</role-link>

</security-role-ref>

<security-role-ref>

   <description>This ref declares the Anonymous role for this
    bean</description>

   <role-name>AnonymousRole</role-name>
   <role-link>AnonymousRole</role-link>

</security-role-ref>
```

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

Assembly Descriptor

Within the `ebusiness.jar` file, the `ejb-jar.xml` deployment descriptor registers the E-business stateless session and entity JavaBeans with the application assembly descriptor shown in Listing 3-18.

Listing 3-18 Assembly Descriptor Element for the E-business JavaBeans

```
<assembly-descriptor>

      <security-role>
         <description>Registered customers</description>
         <role-name>CustomerRole</role-name>
      </security-role>

      <security-role>
         <description>Administrators</description>
         <role-name>AdministrativeRole</role-name>
      </security-role>

      <security-role>
         <description>Anonymous Users</description>
         <role-name>AnonymousRole</role-name>
      </security-role>

    <method-permission>

       <role-name>AnonymousRole</role-name>

       <!-- com.beasys.commerce.ebusiness.catalog.CatalogManager
       -->

       <method>
          <ejb-name>com.beasys.commerce.ebusiness.catalog.
          CatalogManager</ejb-name>
          <method-name>getCategoryManager</method-name>
       </method>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.catalog.
          CatalogManager</ejb-name>
          <method-name>getProductItemManager</method-name>
      </method>

      <method>
         <ejb-name>com.beasys.commerce.ebusiness.catalog.
```

```
   CatalogManager</ejb-name>
  <method-name>getCatalogQueryManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.
   CatalogManager</ejb-name>
   <method-name>getCustomDataManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.
   CatalogManager</ejb-name>
   <method-name>createCatalogRequest</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.
   CatalogManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.category.
 CategoryManager  -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getParent</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getAncestors</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getRootCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSubCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSubCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSiblings</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSiblingKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
```

```
   category.CategoryManager</ejb-name>
   <method-name>getCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSubCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getSiblingCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-name>getItemCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 ProductItemManager -->
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getKeywords</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-name>getItem</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.ProductItemManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 JdbcProductItemManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
```

```
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
  <method-name>getKeywords</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
   <method-name>getItem</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.data.
 CustomDataManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   data.CustomDataManager</ejb-name>
   <method-name>*</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   data.CustomDataManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.query.
 CatalogQueryManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   query.CatalogQueryManager</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   query.CatalogQueryManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.data.
 EpmCustomDataManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   data.EpmCustomDataManager</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   data.EpmCustomDataManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.query.
 JdbcCatalogQueryManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   query.JdbcCatalogQueryManager</ejb-name>
   <method-name>*</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   query.JdbcCatalogQueryManager</ejb-name>
  <method-intf>Home</method-intf>
  <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.category.
 JdbcCategoryManager  -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>

  <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.JdbcCategoryManager</ejb-name>
  <method-name>getParent</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getAncestors</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getRootCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
```

```
    <method-name>getCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblings</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblingKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategoryCount</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblingCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.JdbcCategoryManager</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.security.
    Encryptor</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.tax.taxware.
    TaxwareTaxCalculator</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.customer.
    Customer</ejb-name>
   <method-name>*</method-name>
</method>
```

```
        <method>
            <ejb-name>com.beasys.commerce.ebusiness.shipping.
             ShippingMethod</ejb-name>
            <method-name>*</method-name>
        </method>

        <method>
            <ejb-name>com.beasys.commerce.ebusiness.shoppingcart.
             pipeline.RefreshSavedListPC</ejb-name>
            <method-name>*</method-name>
        </method>

    </method-permission>


    <method-permission>

        <role-name>CustomerRole</role-name>

        <!-- com.beasys.commerce.ebusiness.catalog.
         CatalogManager -->

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.
            CatalogManager</ejb-name>
           <method-name>getCategoryManager</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.
            CatalogManager</ejb-name>
           <method-name>getProductItemManager</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.
            CatalogManager</ejb-name>
           <method-name>getCatalogQueryManager</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.
            CatalogManager</ejb-name>
           <method-name>getCustomDataManager</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.
            CatalogManager</ejb-name>
           <method-name>createCatalogRequest</method-name>
        </method>
```

```
<!-- com.beasys.commerce.ebusiness.catalog.service.category.
 CategoryManager  -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getParent</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getAncestors</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getRootCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getCategoryCount</method-name>
</method>

<method>
  <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
```

```
  <method-name>getSubCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getSubCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getSiblings</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
  <method-name>getSiblingKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
  <method-name>getCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getSubCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getSiblingCount</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.CategoryManager</ejb-name>
   <method-name>getItemCategories</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 ProductItemManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>getKeywords</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
```

```
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>getItem</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 JdbcProductItemManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>getKeywords</method-name>
</method>

<method>
  <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   item.JdbcProductItemManager</ejb-name>
  <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>getItem</method-name>
</method>
```

```
<!-- com.beasys.commerce.ebusiness.catalog.service.
 data.CustomDataManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    data.CustomDataManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.
 query.CatalogQueryManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    query.CatalogQueryManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.data.
 EpmCustomDataManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    data.EpmCustomDataManager</ejb-name>
    <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.
 query.JdbcCatalogQueryManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    query.JdbcCatalogQueryManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.
 category.JdbcCategoryManager  -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCatalogManager</method-name>
</method>
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItems</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getParent</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getAncestors</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getRootCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
```

```
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblings</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblingKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategoryKeys</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSubCategoryCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getSiblingCount</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getCategory</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
```

```
   <method-name>getCategories</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>getItemCategories</method-name>
</method>


<method>
   <ejb-name>com.beasys.commerce.ebusiness.security.
    Decryptor</ejb-name>
   <method-name>*</method-name>
</method>


<method>
   <ejb-name>com.beasys.commerce.ebusiness.payment.
    CreditCardService</ejb-name>
   <method-name>*</method-name>
</method>


<method>
   <ejb-name>com.beasys.commerce.ebusiness.payment.
    PaymentTransaction</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.shoppingcart.
    pipeline.DeleteProductItemFromSavedListPC</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.shoppingcart.
    pipeline.RefreshSavedListPC</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.shoppingcart.
    pipeline.MoveProductItemToSavedListPC</ejb-name>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.beasys.commerce.ebusiness.shoppingcart.
```

```
            pipeline.MoveProductItemToShoppingCartPC</ejb-name>
          <method-name>*</method-name>
      </method>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.order.
           Order</ejb-name>
          <method-name>*</method-name>
      </method>

      <method>
           <ejb-name>com.beasys.commerce.ebusiness.order.
            OrderManager</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
      </method>

  </method-permission>


  <method-permission>

      <role-name>AdministrativeRole</role-name>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.security.
           Decryptor</ejb-name>
          <method-name>*</method-name>
      </method>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.payment.
           CreditCardService</ejb-name>
          <method-name>*</method-name>
      </method>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.payment.
           PaymentTransaction</ejb-name>
          <method-name>*</method-name>
      </method>

      <method>
          <ejb-name>com.beasys.commerce.ebusiness.order.
           Order</ejb-name>
          <method-name>*</method-name>
      </method>


       <!-- com.beasys.commerce.ebusiness.catalog.CatalogManager
       -->
```

```
<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.
    CatalogManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.
 category.CategoryManager -->

<method>

  <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
   category.CategoryManager</ejb-name>
  <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.
 JdbcCategoryManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    category.JdbcCategoryManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 ProductItemManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.ProductItemManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.item.
 JdbcProductItemManager -->

<method>
   <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
    item.JdbcProductItemManager</ejb-name>
   <method-name>*</method-name>
</method>


<!-- com.beasys.commerce.ebusiness.catalog.service.
 data.CustomDataManager -->
```

```
        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
            data.CustomDataManager</ejb-name>
           <method-name>*</method-name>
        </method>


        <!-- com.beasys.commerce.ebusiness.catalog.service.query.
         CatalogQueryManager -->

        <method>

          <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
           query.CatalogQueryManager</ejb-name>
          <method-name>*</method-name>
        </method>


        <!-- com.beasys.commerce.ebusiness.catalog.service.data.
         EpmCustomDataManager -->

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
            data.EpmCustomDataManager</ejb-name>
           <method-name>*</method-name>
        </method>


        <!-- com.beasys.commerce.ebusiness.catalog.service.query.
         JdbcCatalogQueryManager -->

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
            query.JdbcCatalogQueryManager</ejb-name>
           <method-name>*</method-name>
        </method>

        <method>
           <ejb-name>com.beasys.commerce.ebusiness.order.
            OrderManager</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
        </method>

    </method-permission>

    ...

</assembly-descriptor>
```

Note: The <method> tag without a <method-intf> specifies the <method-name> on both the Home and Remote interfaces.

In this case, the assembly descriptor specifies that users in the AnonymousRole and CustomerRole can access the following methods in the com.beasys.commerce.ebusiness.catalog.CategoryManager JavaBean:

- getCategoryManager
- getProductItemManager
- getCatalogQueryManager
- getCustomDataManager
- CreateCatalogRequest

Note: Users in the AnonymousRole *only* can also access all methods for the Home interface of the CategoryManager JavaBean.

Users in the the AnonymousRole and CustomerRole can also access the following methods in the com.beasys.commerce.ebusiness.catalog.service.category.CategoryManager and com.beasys.commerce.ebusiness.catalog.service.category.JdbcCategoryManager JavaBeans:

- getCatalogManager
- getItems
- getItemKeys
- getParent
- getAncestors
- getRootCategory
- getCategoryCount
- getSubCategories
- getSubCategoryKeys
- getSiblings
- getSiblingKeys
- getCategories
- getCategoryKeys
- getItemCount

- `getSubCategoryCount`
- `getSiblingCount`
- `getCategory`
- `getCategories`
- `getItemCategories`

Note: Users in the `AnonymousRole` *only* can also access all methods for the `Home` interface of the `CategoryManager` and `JdbcCategoryManager` JavaBeans.

Users in the `AnonymousRole` and `CustomerRole` can also access the following methods of the `com.beasys.commerce.ebusiness.catalog.service.item.ProductItemManager` and `com.beasys.commerce.ebusiness.catalog.service.item.JdbcProductItemManager` JavaBeans:

- `getCatalogManager`
- `getItemCount`
- `getKeywords`
- `getItemKeys`
- `getItems`
- `getItem`

Note: Users in the `AnonymousRole` *only* can also access all methods for the `Home` interface of the `ProductItemManager` and `JdbcProductItemManager` JavaBeans.

Users in the `AnonymousRole` can also access all methods in the `Home` and `Remote` interfaces for the following JavaBeans:

- `com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManager`
- `com.beasys.commerce.ebusiness.catalog.query.CatalogQueryManager`
- `com.beasys.commerce.ebusiness.catalog.service.data.EpmCustomDataManager`
- `com.beasys.commerce.ebusiness.catalog.service.query.JdbcCatalogQueryManager`
- `com.beasys.commerce.ebusiness.security.Encryptor`
- `com.beasys.commerce.ebusiness.tax.taxware.TaxWareTaxCalculator`

- com.beasys.commerce.ebusiness.customer.Customer
- com.beasys.commerce.ebusiness.shipping.ShippingMethod
- com.beasys.commerce.ebusiness.shoppingcart.pipeline.RefreshSavedListPC

Users in the CustomerRole can also access all methods in the Home and Remote interfaces for the following JavaBeans:

- com.beasys.commerce.ebusiness.catalog.service.data.CustomDataManager
- com.beasys.commerce.ebusiness.catalog.query.CatalogQueryManager
- com.beasys.commerce.ebusiness.catalog.service.data.EpmCustomDataManager
- com.beasys.commerce.ebusiness.catalog.service.query.JdbcCatalogQueryManager
- com.beasys.commerce.ebusiness.security.Decryptor
- com.beasys.commerce.ebusiness.payment.CreditCardService
- com.beasys.commerce.ebusiness.payment.PaymentTransaction
- com.beasys.commerce.ebusiness.shoppingcart.pipeline.DeleteProductItemFromSavedListPC
- com.beasys.commerce.ebusiness.shoppingcart.pipeline.RefreshSavedListPC
- com.beasys.commerce.ebusiness.shoppingcart.pipeline MoveProductItemToSavedListPC
- com.beasys.commerce.ebusiness.shoppingcart.pipeline MoveProductItemToShoppingCartPC
- com.beasys.commerce.ebusiness.order.Order

Lastly, users in the CustomerRole can access all Home methods for the com.beasys.commerce.ebusiness.order.OrderManager JavaBean.

For users in the AdministrativeRole, access to all methods in the Home and Remote interfaces is granted for each of the E-business JavaBeans, excluding com.beasys.commerce.ebusiness.order.OrderManager, to which users are only granted access to the methods in the Home interface.

Security-Role Assignments

Within the `ebusiness.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the E-business stateless session and entity JavaBeans, as shown in Listing 3-19. In this case, the security role `CustomerRole` is assigned to the `wlcs_customer` principal, the `AdministrativeRole` to the `admin` principal, and the `AnonymousRole` to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-19 Security Role Assignments for the E-business JavaBeans

```
<security-role-assignment>

   <role-name>CustomerRole</role-name>
   <principal-name>wlcs_customer</principal-name>

</security-role-assignment>

<security-role-assignment>

    <role-name>AdministrativeRole</role-name>
    <principal-name>admin</principal-name>

</security-role-assignment>

<security-role-assignment>

    <role-name>AnonymousRole</role-name>
    <principal-name>everyone</principal-name>

</security-role-assignment>
```

ejbadvisor.jar

The `ejbadvisor.jar` file contains an EJB that provides the personalization Advisor and advislet framework. For more information about the Advisor, see "Creating Personalized Applications with the Advisor" in the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-8 lists the enterprise JavaBean that is defined within the `ejbadvisor.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-8 ejbadvisor.jar ejb-jar.xml EJB Definition

Name	Type
`com.bea.commerce.platform.advisor.EjbAdvisor`	`Stateless session`

Assembly Descriptor

Within the `ejbadvisor.jar` file, the `ejb-jar.xml` deployment descriptor registers the `EjbAdvisor` stateless session JavaBean with the application assembly descriptor shown in Listing 3-20.

Listing 3-20 Assembly Descriptor Element for the EjbAdvisor JavaBean

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

      <role-name>AnonymousRole</role-name>
```

```
      <method>
         <ejb-name>com.bea.commerce.platform.advisor.
          EjbAdvisor</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

   </method-permission>

 ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `EjbAdvisor` JavaBean's `Home` interface.

Security-Role Assignments

Within the `ejbadvisor.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `EjbAdvisor` stateless session JavaBean, as shown in Listing 3-21. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-21 Security Role Assignments for the EjbAdvisor JavaBean

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

events.jar

The `events.jar` file contains an EJB that provides the Event Service. For more information about the Event Service, see the *Events and Behavior Tracking* documentation.

Enterprise Bean Definitions

Table 3-9 lists the enterprise JavaBean that is defined within the `events.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-9 events.jar ejb-jar.xml EJB Definition

Name	Type
`EventService`	`Stateless session`

Assembly Descriptor

Within the `events.jar` file, the `ejb-jar.xml` deployment descriptor registers the `EventService` stateless session JavaBean with the application assembly descriptor shown in Listing 3-22.

Listing 3-22 Assembly Descriptor Element for the EventService JavaBean

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

      <role-name>AnonymousRole</role-name>
```

```
      <method>
         <ejb-name>com.bea.commerce.platform.advisor.
          EventService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

   </method-permission>

 ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `EventService` JavaBean's `Home` interface.

Security-Role Assignments

Within the `events.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `EventService` stateless session JavaBean, as shown in Listing 3-23. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-23 Security Role Assignments for the EventService JavaBean

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

foundation.jar

The `foundation.jar` file is a collection of EJBs that provides the property set definitions, the property manager, and the Pipeline execution service. For more informaion about property sets and property set management, see "Creating and Managing Property Sets" in the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-10 lists the enterprise JavaBeans that are defined within the `events.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-10 foundation.jar ejb-jar.xml EJB Definitions

Name	Type
`com.beasys.commerce.foundation.property.Schema`	`Entity`
`com.beasys.commerce.foundation.property.EntityPropertyAggregator`	`Stateless session`
`com.beasys.commerce.ebusiness.catalog.service.data.EntityPropertyManager`	`Stateless session`
`com.beasys.commerce.foundation.property.EntityPropertyManager`	`Stateless session`
`com.beasys.commerce.foundation.property.LDAPEntityPropertyManager`	`Stateless session`
`com.beasys.commerce.foundation.property.DirectPropertyManager`	`Stateless session`
`com.beasys.commerce.foundation.property.SchemaManager`	`Stateless session`
`com.beasys.commerce.foundation.pipeline.PipelineExecutor`	`Stateless session`

Assembly Descriptor

Within the `foundation.jar` file, the `ejb-jar.xml` deployment descriptor registers the Foundation stateless session and entity JavaBeans with the application assembly descriptor shown in Listing 3-24.

Listing 3-24 Assembly Descriptor Element for the Foundation JavaBeans

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

      <role-name>AnonymousRole</role-name>

       <method>
          <ejb-name>com.beasys.commerce.foundation.property.
           EntityPropertyAggregator</ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name>
       </method>

   </method-permission>

   <method-permission>

      <role-name>AnonymousRole</role-name>

      <method>
         <ejb-name>com.beasys.commerce.ebusiness.catalog.service.
          data.EntityPropertyManager</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

   </method-permission>

   <method-permission>

      <role-name>AnonymousRole</role-name>

      <method>
         <ejb-name>com.beasys.commerce.foundation.property.
```

```
            EntityPropertyManager</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
        </method>

    </method-permission>

    <method-permission>

        <role-name>AnonymousRole</role-name>

        <method>
            <ejb-name>com.beasys.commerce.foundation.property.
             LDAPEntityPropertyManager</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
        </method>

    </method-permission>

    <method-permission>

        <role-name>AnonymousRole</role-name>

        <method>
            <ejb-name>com.beasys.commerce.foundation.property.
             DirectPropertyManager</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
        </method>

    </method-permission>

    <method-permission>

        <role-name>AnonymousRole</role-name>

        <method>
            <ejb-name>com.beasys.commerce.foundation.property.
             SchemaManager</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
        </method>

    </method-permission>

    <method-permission>

        <role-name>AnonymousRole</role-name>

        <method>
            <ejb-name>com.beasys.commerce.foundation.pipeline.
```

```
            PipelineExecutor</ejb-name>
           <method-intf>Home</method-intf>
           <method-name>*</method-name>
        </method>

    </method-permission>

 ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the each of the Foundation JavaBeans' `Home` interfaces.

Security-Role Assignments

Within the `foundation.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the Foundation stateless session and entity JavaBeans, as shown in Listing 3-25. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-25 Security Role Assignments for the Foundation JavaBeans

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>
```

mail.jar

The `mail.jar` file contains an EJB that provides the outbound Mail Service. The Mail Service uses the JavaMail API to send campaign-related messages to customers, in batches. For more information about the Mail Service, see "How Campaigns Use the Mail Service" in the *Developing Campaign Infrastructure* documentation.

Enterprise Bean Definitions

Table 3-11 lists the enterprise JavaBean that is defined within the `mail.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-11 mail.jar ejb-jar.xml EJB Definition

Name	Type
`MailService`	`Stateless session`

Assembly Descriptor

Within the `mail.jar` file, the `ejb-jar.xml` deployment descriptor registers the `MailService` stateless session JavaBean with the application assembly descriptor shown in Listing 3-26.

Listing 3-26 Assembly Descriptor Element for the MailService JavaBean

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <security-role>
       <description>Administrative Users</description>
```

```
    <role-name>AdminRole</role-name>
</security-role>

<method-permission>

   <role-name>AnonymousRole</role-name>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Home</method-intf>
     <method-name>*</method-name>
   </method>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>sendMail</method-name>
   </method>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>addToBatch</method-name>
   </method>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>getTextFromJSP</method-name>
   </method>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>getJSPResults</method-name>
   </method>

</method-permission>

<!-- Permissions for the AdminRole -->

<method-permission>

   <role-name>AdminRole</role-name>

   <method>
     <ejb-name>MailService</ejb-name>
     <method-intf>Home</method-intf>
     <method-name>*</method-name>
   </method>
```

```
      <method>
        <ejb-name>MailService</ejb-name>
        <method-intf>Remote</method-intf>
        <method-name>*</method-name>
      </method>

   </method-permission>

  ...
</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `MailService` JavaBean's `Home` interface, and the following methods in the `Remote` interface:

- `sendMail`
- `addToBatch`
- `getTextFromJSP`
- `getJSPResults`

Further, users within the security role `AdminRole` will be granted access to all the methods in the `MailService` JavaBean's `Home` and `Remote` interfaces.

Security-Role Assignments

Within the `mail.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `MailService` stateless session JavaBean, as shown in Listing 3-27. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal, and the `AdminRole` is assigned to the `admin` and `system` principals.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-27 Security Role Assignments for the MailService JavaBean

```
<security-role-assignment>
```

```
    <role-name>AnonymousRole</role-name>
    <principal-name>everyone</principal-name>

</security-role-assignment>

<security-role-assignment>

     <role-name>AdminRole</role-name>
     <principal-name>admin</principal-name>
     <principal-name>system</principal-name>

</security-role-assignment>
```

placeholder.jar

The `placeholder.jar` file is a collection of EJBs that provide the placeholder, ads, and ad bucket services. For more information, see "Working with Ad Placeholders" in the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-12 lists the enterprise JavaBeans that are defined within the `placeholder.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-12 placeholder.jar ejb-jar.xml EJB Definitions

Name	Type
`com.bea.commerce.platform.placeholder.PlaceholderService`	`Stateless session`
`com.bea.commerce.platform.ad.AdBucketService`	`Stateless session`
`com.bea.commerce.platform.ad.AdConflictResolver`	`Stateless session`
`com.bea.commerce.platform.ad.AdService`	`Stateless session`

Assembly Descriptor

Within the `placeholder.jar` file, the `ejb-jar.xml` deployment descriptor registers the Placeholder stateless session JavaBeans with the application assembly descriptor shown in Listing 3-28.

Listing 3-28 Assembly Descriptor Element for the Placeholder JavaBeans

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <security-role>
       <description>Administrative Users</description>
       <role-name>AdminRole</role-name>
   </security-role>

   <!-- Permissions for the AnonymousRole -->

   <method-permission>

   <role-name>AnonymousRole</role-name>

      <method>
         <ejb-name>com.bea.commerce.platform.placeholder.
          PlaceholderService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.placeholder.
          PlaceholderService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>getContent</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.placeholder.
          PlaceholderService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>previewContent</method-name>
      </method>
```

```
<method>
   <ejb-name>com.bea.commerce.platform.placeholder.
    PlaceholderService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>setPreviewSlot</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.platform.placeholder.
    PlaceholderService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>removePreviewSlot</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.platform.ad.
    AdBucketService</ejb-name>
   <method-intf>Home</method-intf>
   <method-name>*</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.platform.ad.
    AdBucketService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>getContent</method-name>
</method>

<method>
  <ejb-name>com.bea.commerce.platform.ad.
   AdBucketService</ejb-name>
  <method-intf>Remote</method-intf>
  <method-name>userAddAd</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.platform.ad.
    AdBucketService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>globalAddAd</method-name>
</method>

<method>
   <ejb-name>com.bea.commerce.platform.ad.
    AdBucketService</ejb-name>
   <method-intf>Remote</method-intf>
   <method-name>previewContent</method-name>
</method>
```

```
      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdConflictResolver</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdConflictResolver</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

   </method-permission>


   <!-- Permissions for the AdminRole -->

   <method-permission>

      <role-name>AdminRole</role-name>

      <method>
         <ejb-name>com.bea.commerce.platform.placeholder.
          PlaceholderService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.placeholder.
          PlaceholderService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>
```

```
      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdBucketService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdBucketService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdConflictResolver</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdConflictResolver</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdService</ejb-name>
         <method-intf>Home</method-intf>
         <method-name>*</method-name>
      </method>

      <method>
         <ejb-name>com.bea.commerce.platform.ad.
          AdService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>*</method-name>
      </method>

   </method-permission>

 ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role AnonymousRole will be granted access to all the methods in each of the Placeholder JavaBeans' Home interface. The com.beasys.commerce.platform.ad.AdConflictResolver and com.beasys.commerce.platform.ad.AdService JavaBeans also give users in the AnonymousRole access to the methods in their Remote interfaces.

The com.beasys.commerce.platform.placeholder.PlaceholderService JavaBean allows users in the AnonymousRole to access the following methods in its Remote interface:

- getContent
- previewContent
- setPreviewSlot
- removePreviewSlot

The com.beasys.commerce.platform.ad.AdBucketService JavaBean allows users in the AnonymousRole to access the following methods in its Remote interface:

- getContent
- userAddAd
- globalAddAd
- previewContent

Further, users within the security role AdminRole will be granted access to all the methods in each of the Placeholder JavaBeans' Home and Remote interfaces.

Security-Role Assignments

Within the placeholder.jar file, the weblogic-ejb-jar.xml deployment descriptor defines the security role assignments for the Placeholder stateless session JavaBeans, as shown in Listing 3-29. In this case, the security role AnonymousRole is assigned to the everyone principal, and the AdminRole is assigned to the admin and system principals.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-29 Security Role Assignments for the Placeholder JavaBeans

```
<security-role-assignment>

   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>

</security-role-assignment>

<security-role-assignment>

    <role-name>AdminRole</role-name>
    <principal-name>admin</principal-name>
    <principal-name>system</principal-name>

</security-role-assignment>
```

portal.jar

The `portal.jar` file is a collection of EJBs that provide the portal server.

Enterprise Bean Definitions

Table 3-13 lists the enterprise JavaBeans that are defined within the `portal.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-13 portal.jar ejb-jar.xml EJB Definitions

Name	Type
`ejb.portal.Category`	`Entity`
`ejb.portal.ColumnInformation`	`Entity`
`ejb.portal.GroupPersonalization`	`Entity`
`ejb.portal.PortalDefinition`	`Entity`

Table 3-13 portal.jar ejb-jar.xml EJB Definitions

Name	Type
ejb.portal.PortalGroupHierarchy	Entity
ejb.portal.PortalHierarchy	Entity
ejb.portal.PortalManager	Stateless session
ejb.portal.PortalPersonalization	Entity
ejb.portal.PortletDefinition	Entity
ejb.portal.UserPersonalization	Entity

Security-Role References

In addition, the `ejb-jar.xml` deployment descriptor contains security-role reference elements for the `ejb.portal.PortalManager` stateless session JavaBean. These elements, shown in Listing 3-30, enable the EJB to do programmatic security checking, if such behavior is desired.

Listing 3-30 Security-Role Reference in PortalManager JavaBean

```
<security-role-ref>
   <description>This ref declares the Anonymous role for this
    bean</description>
   <role-name>AnonymousRole</role-name>
   <role-link>AnonymousRole</role-link>
</security-role-ref>
```

Note: For information on the differences between declarative and programmatic security, see the “Security” chapter in the *Java 2 Platform Enterprise Edition Specification, v1.3*.

Assembly Descriptor

Within the `portal.jar` file, the `ejb-jar.xml` deployment descriptor registers the Portal entity and stateless session JavaBeans with the application assembly descriptor shown in Listing 3-31.

Listing 3-31 Assembly Descriptor Element for the Portal JavaBeans

```
<assembly-descriptor>

   <security-role>
       <description>Anonymous Users</description>
       <role-name>AnonymousRole</role-name>
   </security-role>

   <method-permission>

       <role-name>AnonymousRole</role-name>

       <method>
          <ejb-name>ejb.portal.PortalManager</ejb-name>
          <method-intf>Home</method-intf>
          <method-name>*</method-name
       </method>

   </method-permission>

 ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `ejb.portal.PortalManager` JavaBean's `Home` interface.

Security-Role Assignments

Within the `portal.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the Portal stateless session and entity JavaBeans, as shown in Listing 3-32. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-32 Security Role Assignments for the Portal JavaBeans

```
<security-role-assignment>
   <role-name>AnonymousRole</role-name>
   <principal-name>everyone</principal-name>
</security-role-assignment>
```

priceService.jar

The `priceService.jar` file contains an EJB that provides the dynamic product pricing engine. For more information, see "Price Service" in the Discounts chapter of the *Managing Purchases and Processing Orders* documentation.

Enterprise Bean Definitions

Table 3-14 lists the enterprise JavaBean that is defined within the `priceService.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-14 priceService.jar ejb-jar.xml EJB Definition

Name	Type
PriceService	Stateless session

Assembly Descriptor

Within the priceService.jar file, the ejb-jar.xml deployment descriptor does not register the PriceService stateless session JavaBean with an application assembly descriptor, nor does it define security roles or grant permissions on the JavaBean's methods. This is because PriceService is an EJB that had no security constraints.

Note: Other EJBs that have no security constraints may still have content within the <assembly-descriptor> element. These EJBs were written for a prior release of the BEA WebLogic Server, which required a <method> element for the Home interface even if all methods in the EJB were open.

Security-Role Assignments

Because there are no security contraints on the priceService stateless session JavaBean, the weblogic-ejb-jar.xml deployment descriptor does not define any security role assignments for it.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

rules.jar

The rules.jar file contains an EJB that provides the Rules Manager, which is the public interface to the rules engine. For more information, see "Introducing the Rules Manager" in the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-15 lists the enterprise JavaBean that is defined within the `rules.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-15 rules.jar ejb-jar.xml EJB Definition

Name	Type
`com.bea.commerce.platform.rules.manager.RulesManager`	`Stateless session`

Assembly Descriptor

Within the `rules.jar` file, the `ejb-jar.xml` deployment descriptor registers the `RulesManager` stateless session JavaBean with the application assembly descriptor shown in Listing 3-33.

Listing 3-33 Assembly Descriptor Element for the RulesManager JavaBean

```
<assembly-descriptor>

    <security-role>
      <description>Administrative Users</description>
      <role-name>AdministrativeRole</role-name>
    </security-role>

    <security-role>
      <description>Rule Reading Users</description>
      <role-name>RulesReaderRole</role-name>
    </security-role>

    <security-role>
      <description>Anonymous Users</description>
      <role-name>AnonymousRole</role-name>
    </security-role>

    <method-permission>

      <role-name>AnonymousRole</role-name>

      <method>
         <ejb-name>com.bea.commerce.platform.rules.manager.
```

```
      RulesManager</ejb-name>
      <method-intf>Home</method-intf>
      <method-name>*</method-name>
  </method>

</method-permission>

<method-permission>

  <role-name>AnonymousRole</role-name>

  <method>
     <ejb-name>com.bea.commerce.platform.rules.manager.
      RulesManager</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>evaluateRuleSet</method-name>
  </method>

</method-permission>

<method-permission>

  <role-name>AnonymousRole</role-name>

  <method>
     <ejb-name>com.bea.commerce.platform.rules.manager.
      RulesManager</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>evaluateRule</method-name>
  </method>

</method-permission>

<method-permission>

  <role-name>AdministrativeRole</role-name>

  <method>
     <ejb-name>com.bea.commerce.platform.rules.manager.
      RulesManager</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>setRuleSet</method-name>
  </method>

</method-permission>

<method-permission>

  <role-name>AdministrativeRole</role-name>
  <role-name>RulesReaderRole</role-name>
```

```
        <method>
           <ejb-name>com.bea.commerce.platform.rules.manager.
            RulesManager</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>getRuleSet</method-name>
        </method>

      </method-permission>

      <method-permission>

        <role-name>AdministrativeRole</role-name>

        <method>
           <ejb-name>com.bea.commerce.platform.rules.manager.
            RulesManager</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>getRuleSets</method-name>
        </method>

      </method-permission>

      <method-permission>

        <role-name>AdministrativeRole</role-name>

        <method>
           <ejb-name>com.bea.commerce.platform.rules.manager.
            RulesManager</ejb-name>
           <method-intf>Remote</method-intf>
           <method-name>removeRuleSet</method-name>
        </method>

      </method-permission>

    ...

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `RulesManager` JavaBean's `Home` interface. Additionally, users within the `AnonymousRole` will be granted access to the `evaluateRuleSet` and `evaluateRule` methods in the `Remote` interface.

Users in the AdministrativeRole will be granted access to the setRuleSet, getRuleSet, getRuleSets, and removeRuleSet methods in the RulesManager JavaBean's Remote interface. Users in the RulesReaderRole will also be granted access to the getRuleSet method in the Remote interface.

Security-Role Assignments

Within the rules.jar file, the weblogic-ejb-jar.xml deployment descriptor defines the security role assignments for the RulesManager stateless session bean, as shown in Listing 3-34. In this case, the security role AdministrativeRole is assigned to the admin and system principals, the RulesReaderRole is assigned to the wlcm_internal principal, and the AnonymousRole is assigned to the everyone principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-34 Security Role Assignments for the RulesManager JavaBean

```
<security-role-assignment>

    <role-name>AdministrativeRole</role-name>
    <principal-name>admin</principal-name>
    <principal-name>system</principal-name>

</security-role-assignment>

<security-role-assignment>

    <role-name>RulesReaderRole</role-name>
    <principal-name>wlcm_internal</principal-name>

</security-role-assignment>

<security-role-assignment>

    <role-name>AnonymousRole</role-name>
    <principal-name>everyone</principal-name>

</security-role-assignment>
```

servicemgr.jar

The servicemgr.jar file contains an EJB that provides version information.

Enterprise Bean Definitions

Table 3-16 lists the enterprise JavaBean that is defined within the servicemgr.jar file's ejb-jar.xml deployment descriptor.

Table 3-16 servicemgr.jar ejb-jar.xml EJB Definition

Name	Type
com.beasys.commerce.servicemanager.CommerceServiceManager	Stateless session

Assembly Descriptor

Within the servicemgr.jar file, the ejb-jar.xml deployment descriptor registers the CommerceServiceManager stateless session JavaBean with the application assembly descriptor shown in Listing 3-35.

Listing 3-35 Assembly Descriptor Element for the CommerceServiceManager JavaBean

```
<assembly-descriptor>

      <security-role>
        <description>Anonymous Users</description>
        <role-name>AnonymousRole</role-name>
      </security-role>

    <method-permission>

          <role-name>AnonymousRole</role-name>
```

```
            <method>
               <ejb-name>com.beasys.commerce.servicemanager.
                CommerceServiceManager</ejb-name>
               <method-intf>Home</method-intf>
               <method-name>*</method-name>
            </method>

    </method-permission>

</assembly-descriptor>
```

In this case, the assembly descriptor specifies that users within the security role `AnonymousRole` will be granted access to all the methods in the `CommerceServiceManager` JavaBean's `Home` interface.

Security-Role Assignments

Within the `servicemgr.jar` file, the `weblogic-ejb-jar.xml` deployment descriptor defines the security role assignments for the `CommerceServiceManager` stateless session JavaBean, as shown in Listing 3-36. In this case, the security role `AnonymousRole` is assigned to the `everyone` principal.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

Listing 3-36 Security Role Assignments for the CommerceServiceManager JavaBean

```
<security-role-assignment>

    <role-name>AnonymousRole</role-name>
    <principal-name>everyone</principal-name>

</security-role-assignment>
```

uupexample.jar

The `uupexample.jar` file contains an EJB that provides an example of the Unified User Profile (UUP). For more information about UUP, see "Unified User Profiles" in the Creating and Managing Users chapter of the *Building Personalized Applications* documentation.

Enterprise Bean Definitions

Table 3-17 lists the enterprise JavaBean that is defined within the `uupexample.jar` file's `ejb-jar.xml` deployment descriptor.

Table 3-17 uupexample.jar ejb-jar.xml EJB Definition

Name	Type
`examples.usermgmt.UnifiedUser`	`Entity`

Assembly Descriptor

Within the `uupexample.jar` file, the `ejb-jar.xml` deployment descriptor does not register the `UnifiedUser` entity JavaBean with an application assembly descriptor, nor does it define security roles or grant permissions on the JavaBean's methods. This is because `UnifiedUser` is an EJB that had no security constraints.

Note: Other EJBs that have no security constraints may still have content within the `<assembly-descriptor>` element. These EJBs were written for a prior release of the BEA WebLogic Server, which required a `<method>` element for the `Home` interface even if all methods in the EJB were open.

Security-Role Assignments

Because there are no security contraints on the `UnifiedUser` entity JavaBean, the `weblogic-ejb-jar.xml` deployment descriptor does not define any security role assignments for it.

Note: For more information about security roles and principals, see "What Is a Security Role?" on page 2-3.

4 Sample Applications, Administration Tools, and Security

Out-of-the-box, the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server products include sample applications that provide you with a starting point for building your own e-commerce applications, and tools that allow you to effectively administer and manage those applications.

As such, these sample applications and administration tools must occasionally implement J2EE security measures. For example, because the JavaServer Page (JSP) templates included in the WebLogic Commerce Server simulate a real-world storefront (including a check out process), some of the templates must be protected. Similarly, not everyone in an organization should have the ability to make modifications to your critical e-business applications. Thus, security measures are also required and implemented in the included administration tools.

While the inner workings of security in these applications has been addressed in previous topics, this topic describes how a customer's user experience might be affected by security in the sample applications (that is, the WebLogic Commerce Server sample storefront application and the WebLogic Personalization Server example portal application). It also provides details about credit card security measures, and describes how the security implemented in the administration tools may affect administrative users.

This topic includes the following sections:

- Security in the Sample Storefront Application
  - Protected JavaServer Page (JSP) Templates

Security in the Sample Storefront Application

The WebLogic Commerce Server sample storefront 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 catalog. In summary, the JSP templates are a great starting point for building your own e-commerce Web application!

Notes: You can view the sample storefront application by first starting the WebLogic Commerce Server and then selecting Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → WLCS Templates (on Windows systems).

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

The files for the sample storefront application are located in `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/applications/wlcsApp/wlcs` directory, where `$WL_COMMERCE_HOME` is the directory in which you installed the product.

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.

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 WebLogic Commerce Server sample storefront 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. This section describes how these JSP templates are protected, and includes the following subsections:

- main.jsp Template Versions
- Form-based Authentication and Access to Protected Pages
- Session Inactivity
- SSL and Declarative Transport

main.jsp Template Versions

The `main.jsp` template is the default home page for the `wlcs` sample storefront 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" chapter of the *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 are they prompted for a username and password. Customers who chose to log in prior to accessing protected pages and are already authenticated are not prompted again unless they log out or their session times out.

If a user has successfully logged in, that user is considered a registered customer, welcomed with a personalized greeting, and presented with additional links on the secured version of the `main.jsp` template (`secureMain.jsp`).

Form-based Authentication and Access to Protected Pages

If an anonymous user attempts to load a protected page or explicitly indicates their intent to log in (by clicking the Log In link on the `main.jsp` template), the `wlcs` application invokes form-based authentication techniques to present the `login.jsp` template to the user. The top portion of this template provides form fields for the user to enter and submit a previously created username and password combination.

Note: Form-based authentication lets developers 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.

When the user submits their username and password to gain access to their account, the `j_security_check` servlet passes this information 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 correct password is supplied.

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 the `login.jsp` from the `main.jsp` template, the secured version of the `main.jsp` template (`secureMain.jsp`) is simply loaded. 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" chapter of the *Registering Customers and Managing Customer Services* documentation.

While you are viewing the WebLogic Commerce Server sample storefront 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, be sure to purchase and install a real server license from a certificate authority. For more information about server certificates, see "Managing Security" in the *BEA WebLogic Server Administration Guide*.

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 can be customized to something other than the default value of 15 minutes. If you wish to modify this value, see "Session Timeout" on page 2-9.

SSL and Declarative Transport

Lastly, the Webflow and Pipeline mechanisms that direct the presentation and business logic associated with the 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 `wlcs` application. For more information on these topics, see "Links Using HTTPS" on page 2-13.

Secured JSP templates that rely on SSL also require a setting 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, see "Secure JavaServer Pages (JSPs)" on page 2-14.

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 specfic 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 WebLogic Commerce Server product's 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.

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.

Notes: For more information about RSA security standards, see http://www.rsasecurity.com/. For more technical information about the Credit Card Security Service, please contact your BEA representative.

Customizable Security Settings

Although the WebLogic Commerce Server product specifies default settings for the Credit Card Security Service, you can customize them. The security settings reside in the `weblogiccommerce.properties` file (located in the `$WL_COMMERCE_HOME` directory, where `$WL_COMMERCE_HOME` is the directory in which you installed WebLogic Commerce Server). These security settings are shown in Listing 4-1.

Listing 4-1 Security Settings in weblogiccommerce.properties

```
#####################################################
# Properties required for the Security Service
#####################################################

# Security services are turned on by setting this property to true.
# Commenting out the property or setting it to false will disable
# security.

is.encryption.enabled=true


# The name of the security table and column names for the public
# and private 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
```

```
# WARNING! Remember that setting this property will start up the
# server without prompting for a password. The password will be read
# from this property which makes the encryption vulnerable to an
# inside attack.

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, WebLogic Commerce Server ships with a 1024-bit key size, although you may want to modify the encryption strength by changing the key size. Table 4-1 illustrates the possible key bit values.

Table 4-1 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 4-9.

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 `$WL_COMMERCE_HOME` directory, where `$WL_COMMERCE_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 information on setting up a startup class, see "Startup Class" in the *WebLogic Server 6.0* documentation.

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 `$WL_COMMERCE_HOME` directory, where `$WL_COMMERCE_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.

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.
2. Point your Web browser to `<hostname>:port/tools/security/security_getPassword.html`, to load the secure Web form shown in Figure 4-1.

Figure 4-1 security_getPassword.html

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 `$WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp/tools/Web-inf` directory, where `$WL_COMMERCE_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 Administration Tool to reauthorize the transaction. For more information about using the Payment Management Administration Tool, see "Using the Order and Payment Management Pages" in the *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 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.

Security in the Example Portal Application

The WebLogic Personalization Server example portal application includes a set of JavaServer Page (JSP) templates that provide a model for your customers' portal pages. 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 customers to select and arrange the portlets that appear within their portals, allowing for truly personalized information display.

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 default content of your portal. Like the WebLogic Commerce Server (storefront) JSP templates, the WebLogic Personalization Server (portal) JSP templates are a great starting point for building your own portal Web application!

Notes: You can view the example portal application by first starting the WebLogic Commerce Server and then selecting Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → Example Portal (on Windows systems).

The default username/password combination for the `exampleportal` templates is `acme` and `acme`.

The files for the example portal application are located in `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/applications/wlcsApp/exampleportal` directory, where `$WL_COMMERCE_HOME` is the directory in which you installed the product.

Because some portal Web pages display customized content or allow registered users to modify personal information (such as preferences or username and password), 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.

The portal.jsp Template

The `portal.jsp` template is the default home page for the `exampleportal` application; it initially allows all anonymous users to access it. The portal anonymous users are presented with is a default portal. In other words, it contains the portlets, color schemes, organization, content, and so on that are defined for the default portal, and is not currently customized to the user's own preferences.

Because the information contained within a portal page is generated based on the preferences of an individual user, however, the WebLogic Personalization Server example portal application requires that users be authenticated in order to fully use the application.

The `portal.jsp` template does not automatically prompt the user for a username and password using basic or form-based authentication techniques; it requires the user to initiate the login process by clicking the appropriate link/button. Although not fully declarative, the `portal.jsp` template is login-aware. Therefore, the `portal.jsp` template behaves differently for anonymous and authenticated users. Once the portal user has successfully logged in, that user is considered a registered user, welcomed with a personalized greeting, and presented with additional links on the personalized version of the `portal.jsp` template.

Logging Into the Portal

The example portal application contains a number of JSPs that handle user login and delegate authentication activities to the WebLogic Server.

When a user clicks the Sign In link available from the `portal.jsp` template, the following occurs:

- The `_userlogin` JSP is called.

  This JavaServer Page is protected by a security constraint defined in the example portal Web application's `web.xml` deployment descriptor. This constraint causes form-based authentication (via the `login.jsp` template) to be invoked prior to this page being executed.

- The `login.jsp` template is displayed to the portal user.

The form displayed when a portal user needs to enter their username and password is the `login.jsp` template. The `login.jsp` template is written in a non-portal specific manner so it can be reused as is when copying the example portal directory to create a new portal.

- Upon submission of the `login.jsp` template, the `loginRequest` JSP is invoked.

  The `loginRequest` JSP, which wraps both the `_userlogin.jsp` and `login.jsp` pages, is invoked by the servlet container to delegate form-based authentication activities to the WebLogic Server (via the `j_security_check` servlet). The `loginRequest` JSP is also configured using the Web application's `web.xml` deployment descriptor.

- If the authentication was successful, the `loginSuccess.jsp` template is displayed to the portal user.

  Following authentication, control is passed back to the `_userlogin` JSP, which defined the JSP template where the user should be directed once user authentication has completed successfully by the `j_security_check` servlet (that is, `loginSuccess.jsp`). If the portal user is a member of multiple groups, this page allows the user to select a group. Otherwise, it simply forwards the user to the `portal.jsp` template.

A Prerequisite for Logging In

An important point to note about the portal user login process is that in order for a portal user to arrive at the login page (`_userlogin.jsp`), the user must be a member of the `AcmeUsers` group. This is because of the definitions in the `exampleportal` application's `web.xml` deployment descriptor. However, by coming to the `portal.jsp` template, the portal user is automatically considered a member of the `AcmeUsers` group.

Security in the Administration Tools

In addition to the sample storefront and portal applications, the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server also include some tools that members of your organization will use to administer your e-commerce Web site. Presently, these tools are available in two forms: the Administration JSP application (accessed via a Web browser) and the stand-alone E-Business Control Center graphical user interface (GUI) application.

Because both of these tools allow only authorized individuals in your organization to make modifications to your e-commerce Web site, the administration tools also require some security protections. The security implemented in both types of administration tools are discussed in the following sections:

- The Administration JSPs
- The E-Business Control Center

The Administration JSPs

The purpose of the JSP-based Administration Tool is to provide administrative users of the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server with the ability to manage various aspects of an e-commerce Web application. The Administration Tool includes JSPs for User, Property Set, Portal, Content, Webflow/Pipeline, Catalog, Order, and Payment Management. Thus, access to these Administration JSPs must be granted only to authorized users.

Notes: You can view the Administration JSPs by first starting the WebLogic Commerce Server and then selecting Start → Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → Administration Tools (on Windows systems).

The WebLogic Commerce Server defines an `Administrator` username for you, with a password of `password` (lowercase characters).

The files for the Administration Tools application are located in `$WL_COMMERCE_HOME/WebLogicCommerce3.5/config/wlcsDomain/applications/wlcsApp/tools` directory, where `$WL_COMMERCE_HOME` is the directory in which you installed the product.

When users attempt to access the JSP-based Administration Tool, the application invokes basic authentication techniques to present a login screen to the user, as shown in Figure 4-2.

Figure 4-2 Administration Login Screen

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.

When the user submits their username and password to gain access to their account, the `j_security_check` servlet passes this information to the WebLogic Server for verification. The security mechanism in WebLogic Server verifies that the user is part of the `admin` group and that the correct password is supplied.

If the user has supplied a valid username and password combination, WebLogic Server authenticates the user and grants access to the Administration JSPs. Presently, users who have been successfully authenticated by the WebLogic Server have permission to use all the features available in the Administration JSP tools. In other words, authenticated users are “superusers;” the application permits them to make all possible modifications, and there are no restrictions based on the user’s organizational role.

The E-Business Control Center

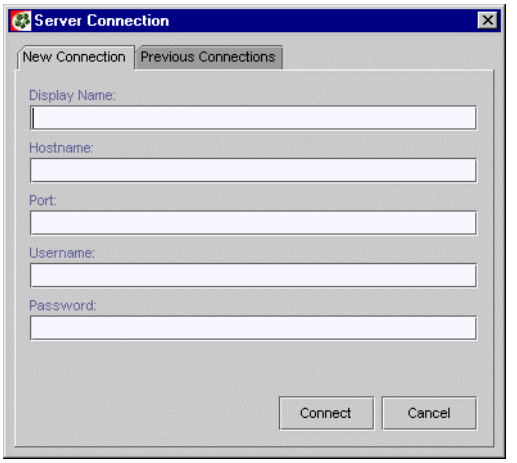
The E-Business Control Center is an application that is packaged with the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, and is designed to simplify the tasks that are necessary to create and maintain a truly personalized Web site. To meet this objective, 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 Administration JSPs in that it allows certain, privileged users to affect the content and behavior of a Web site. However, the similarities end there. First, the E-Business Control Center is a stand-alone application that is started like the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server applications (that is, the E-Business Control Center is not started within a Web browser). Second, the E-Business Control Center is built using Java Swing components and the NetBeans IDE, rather than JavaServer Pages. As such, the appearance of the interface can be better customized and the amount of native (non-Java) code used in the application can be minimized.

Note: To learn more about Java Swing components or the NetBeans IDE, see the Java Tutorial "Getting Started with Swing" or the NetBeans Web site at http://www.netbeans.org, respectively.

To make modifications to a Web site using the E-Business Control Center, a user must connect to a server by providing some connection information, including a username and password combination. One of the E-Business Control Center's Server Connection windows is shown in Figure 4-3.

Figure 4-3 Server Connection Window



Note: For more information about the Server Connection window and the information required to connect, see "Connecting the E-Business Control Center to a Server" in the *Using the E-Business Control Center* documentation.

When a user enters their information into the Server Connection window, the E-Business Control Center uses WebLogic Server realm authentication to create a JDNI context, which it then uses to look up Enterprise JavaBeans (EJBs) on the server side. For more information about WebLogic Server security realms and authentication, see "Security Realms" and other, related topics in the "Security Fundamentals" section of the *Programming WebLogic Security* documentation.

Presently, users who have been successfully authenticated by the WebLogic Server have permission to use all the features available in the E-Business Control Center. In other words, authenticated users are "superusers;" the application permits them to make all possible modifications, and there are no restrictions based on the user's organizational role.

Note: The features available to authenticated 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 Does the E-Business Control Center Provide" in the Introduction of the *Using the E-Business Control Center* documentation.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

P

R

S

T

U

V

W

X