A decorative graphic in the top left corner consisting of a cluster of small squares in various colors including purple, blue, green, yellow, and grey, arranged in a roughly triangular shape.

Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager

Version 4.7
May 31, 2007

ORACLE®

Copyright © 1996, 2007 Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

PRODUCT MODULES AND OPTIONS. This guide contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel's Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

1 Preface

About Customer Self-Service and eaSuite™ 7

About This Guide 8

Related Documentation 8

2 About Oracle Sample Application

Before Getting Started 9

About ear-eStatement.ear 9

Migration Issues 9

About the Samples Directory 9

J2EE Applications (EAR files) and Web Applications (WAR files) 10

eStatement Manager Applications (DDF, ALF, HTML, and XML files) 10

Sample J2EE and Web Applications for eaSuite 11

Sample 11

Training 12

umfsample 12

Sample Datasets for eaSuite 12

About Jobs, Views, and Version Sets 12

3 Deploying and Using Sample

About Sample 15

Deploying the Sample J2EE Web Application 15

Deployment Using WebLogic 16

Deployment Using WebSphere 17

Deployment Using Oracle Application Server 18

Setting up NatlWireless 19

Viewing NatlWireless Statements in Sample 22

4 Renaming Sample to a New J2EE Application

About Application Contexts 27

About the Context Root 27

About JNDI Names 27

About XML Deployment Descriptors 28

Define Your New Context Name 28

Edit EAR File Deployment Descriptors 28

Extract the Sample EAR File 28

Architecture of Sample EAR 29

Edit application.xml 29

Edit EJB Deployment Descriptors 30

About Sample EJBs 30

Extract Descriptors from the EJB JAR File 30

Edit ejb-jar.xml 31

Edit weblogic-ejb-jar.xml (WebLogic) 31

Edit ibm-ejb-jar-bnd.xmi (WebSphere) 32

Repackage the EJB Jar File 33

Repeat for Each EJB in the EAR and WAR Files 33

Edit WAR File Deployment Descriptors 34

Extract the Sample WAR File 34

Architecture of the WAR File 34

Edit web.xml 35

Edit weblogic.xml (WebLogic) 37

Edit ibm-web-bnd.xmi (WebSphere) 38

Package the WAR File 39

Repackage and Deploy the EAR File 39

5 Creating a Custom Web Application for eStatement Manager

Define a Custom Enrollment Model	41
Define Custom Servlets	41
Customize the WAR File	41
Implement JSP Validation	42
About Validation of HTTP Requests	42
Package com.edocs.common.web.validation Description	42
About the ValidatorBean	43
Using ValidatorBean for Parameter Name Checking	44

6 **Appendix A: Components of EAR and WAR Samples**

Components of ear-sample.ear	47
<root> directory of ear-sample.ear	47
/lib directory	48
/meta-inf directory	52
Components of war-sample.war	52
enrollment Directory	52
META-INF Directory	52
user/jsp Directory	52
WEB-INF Directory	52

7 **Appendix B: Components of eStatement Manager Application Datasets**

Components of National Wireless Dataset	55
---	----

Index

About Customer Self-Service and eaSuite™

Oracle has developed the industry's most comprehensive software and services for deploying Customer Self-Service solutions. **eaSuite™** combines electronic presentment and payment (EPP), order management, knowledge management, personalization and application integration technologies to create an integrated, natural starting point for all customer service issues. eaSuite's unique architecture leverages and preserves existing infrastructure and data, and offers unparalleled scalability for the most demanding applications. With deployments across the healthcare, financial services, energy, retail, and communications industries, and the public sector, eaSuite powers some of the world's largest and most demanding customer self-service applications. eaSuite is a standards-based, feature rich, and highly scalable platform, that delivers the lowest total cost of ownership of any self-service solution available.

eaSuite consists of four product families:

- Electronic Presentment and Payment (EPP) Applications
- Advanced Interactivity Applications
- Enterprise Productivity Applications
- Development Tools

Electronic Presentment and Payment (EPP) Applications are the foundation of Oracle's Customer Self-Service solution. They provide the core integration infrastructure between organizations' backend transactional systems and end users, as well as rich e-billing, e-invoicing and e-statement functionality. Designed to meet the rigorous demands of the most technologically advanced organizations, these applications power Customer Self-Service by managing transactional data and by enabling payments and account distribution.

- **eStatement Manager™** is the core infrastructure of enterprise Customer Self-Service solutions for organizations large and small with special emphasis on meeting the needs of organizations with large numbers of customers, high data volumes and extensive integration with systems and business processes across the enterprise. Organizations use eStatement Manager with its data access layer, composition engine, and security, enrollment and logging framework to power complex Customer Self-Service applications.
- **ePayment Manager™** is the electronic payment solution that decreases payment processing costs, accelerates receivables and improves operational efficiency. ePayment Manager is a complete payment scheduling and warehousing system with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments via various payment processing service providers.

Oracle's **Development Tools** are visual development environments for designing and configuring Oracle's Customer Self-Service solutions. The Configuration Tools encompass data and rules

Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manag

management, workflow authoring, systems integration, and a software development kit that makes it easy to create customer and employee-facing self-service applications leveraging eaSuite.

About This Guide

This guide describes the tasks required to deploy and customize the J2EE applications provided by eStatement Manager. It also provides a step by step description of how to deploy the Sample application provided by eStatement Manager, and how to validate that it is set up correctly by running a job through your installed eStatement Manager environment.

Related Documentation

This guide is part of the eStatement Manager documentation set. For more information about implementing your eStatement Manager application, see one of the following guides:

Print Document	Description
<i>Installation and Configuration Guide for Oracle Siebel eStatement Manager</i>	How to install eStatement Manager and configure it in a distributed environment.
<i>Data Definition (DefTool) Guide for Oracle Siebel eStatement Manager</i>	How to create Data Definition Files (DDFs) for use in indexing your application and extracting data for live presentment.
<i>Presentation Design (Composer) Guide for Oracle Siebel eStatement Manager</i>	How to create Application Logic Files (ALFs) to present statement data for dynamic online display.
<i>SDK Guide for Oracle Siebel eStatement Manager</i>	How to use the Oracle Software Developers Kit to write custom code against Oracle applications.

2

About Oracle Sample Application

Before Getting Started

Before you can begin to develop an application for eStatement Manager, you must install and deploy eStatement Manager to your application server using the instructions described in the *Installation and Configuration Guide for Oracle Siebel eStatement Manager* and in the Release Notes. Please verify that eStatement Manager is configured and running correctly before proceeding.

For more specific deployment information about your application server platform, please consult your application server product documentation.

About ear-eStatement.ear

The **ear-eStatement.ear** J2EE application contains the core functionality of eStatement Manager, including the Command Center and the eStatement Manager engine. It also contains placeholder files for other components of the eaSuite.

CAUTION: You should NOT modify the ear-eStatement.ear file, and you should always keep a backup in a safe place.

Migration Issues

eStatement Manager 3.4 and later versions support specifying a data source for each DDN, or eStatement Manager application. The default datasource EJB is **edx/ejb/EdocsDataSource**. When migrating the database from 3.2 or earlier versions to 3.4 or later, you must redeploy **ear-eStatement.ear** (to populate the data source mapping for eStatement Manager) to view statements for eStatement Manager applications created before migration.

For more information about DDN to Datasource mapping, please see the *Presentation Design (Composer) Guide for Oracle Siebel eStatement Manager*.

About the Samples Directory

Your Oracle installation includes the folder `<EDX_HOME>/samples`, which contains J2EE, Web, and eStatement Manager applications for use with the eaSuite. This topic defines each sample and directs you to further information on its use.

In eStatement Manager, the term *application* can refer to three different types of file components: J2EE applications, Web applications, and eStatement Manager applications. Each is defined in the following sections.

J2EE Applications (EAR files) and Web Applications (WAR files)

A *J2EE application* bundles enterprise Java components and services for deployment into a J2EE Server container. The J2EE server provides underlying services to handle transaction and state management, multithreading, resource pooling, and other complex low-level details.

For an introduction to J2EE application components, see

<http://java.sun.com/j2ee/1.4/index.jsp>

J2EE applications are packaged and deployed in an *Enterprise Archive* (EAR) file. An EAR file contains an XML file for its deployment descriptor and one or more EJB .jar and .war files.

- Each EJB JAR file contains a deployment descriptor, the enterprise bean files, and related files.
- Each application client JAR file contains a deployment descriptor, the class files for the application client, and related files.
- Each WAR file contains a deployment descriptor, the Web component files, and related resources.

A *Web application*, sometimes called a *Web app*, is a customized software system of Web services and code components with a business state that is changed by user input, usually through a Web interface. Each unique combination of business features and functionality requires a unique Web application.

Web applications are packaged and deployed in a *Web Application Archive* (WAR) file within the EAR file. The WAR file contains JSPs, servlet classes, HTML templates and image files, EJBs specific to the WAR file, and XML deployment descriptors for each component.

For more information, see [Sample J2EE and Web Applications for eaSuite](#).

eStatement Manager Applications (DDF, ALF, HTML, and XML files)

An *eStatement Manager application* is a customized set of eStatement Manager data files (DDFs, ALFs, HTML templates, and XML/XSLT files) and Command Center jobs, created to extract and present statements online from a particular dataset. Each unique combination of data, business rules, and presentment requires a unique (eStatement Manager) application.

Your eStatement Manager installation includes an example eStatement Manager application dataset called *National Wireless*, a fictional telecommunications company. Your team can use the **NatlWireless** sample files and data with one or more J2EE Web applications to test your installation of eStatement Manager; practice the eStatement Manager toolset including the DefTool, Composer, and Command Center; and customize the sample Web applications for your own environment. For more information about National Wireless, see [Components of National Wireless Dataset](#) and the *Data Definition (DefTool) Guide* and the *Presentation Design (Composer) Guide for Oracle Siebel eStatement Manager*.

Sample J2EE and Web Applications for eaSuite

Installing eStatement Manager and the SDK places enterprise application archives (EAR files) for eStatement Manager and for Sample, Training, reportingSample, and UMFsample (SDK only) in the `<EDX_HOME>/samples` directory of your eStatement Manager installation. Each is described here.

In the installation paths shown, replace `<Sample>` with *Sample*, *Training*, *reportSample* or *umfsample*.

Platform	EAR Installed to:	Deploy EAR to:
WebLogic	<code><EDX_HOME>/samples/<Sample>/J2EEApps/weblogic</code>	<code>WebLogic_home/config/mydomain/applications</code>
WebSphere	<code><EDX_HOME>/samples/<Sample>/J2EEApps/websphere</code>	<code>WebSphere_home/installedApps/</code>
Oracle Application Server	<code><EDX_HOME>/samples/<Sample>/J2EEApps/oracleAS</code>	<code>OAS_home/j2ee/<instance name>/applications/</code>

Sample

The `ear-sample.ear` J2EE application demonstrates the core features of eStatement Manager, including:

- Non-hierarchical enrollment model using Oracle Common Directory Access (CDA)
- Content access to statement summary and detail data
- Line item disputes

Oracle recommends that you use Sample as the skeleton for developing your own custom Web applications. Each SDK module uses Sample to demonstrate how to customize eStatement Manager features.

For a list of the components in Sample, please see [Appendix A: Component Lists for Sample Web Applications](#).

TIP: If you are also using Payment, Oracle recommends customizing one of the Payment Web application EAR files instead of Sample application servers. See Command Center Help for information about bulk publishing.

For more information, see the *SDK Guide for Oracle Siebel eStatement Manager*.

CAUTION: Before modifying any sample EAR file, you should always keep a backup in a safe place.

Training

The **ear-training.ear** J2EE application demonstrates more advanced features of eStatement Manager, including hierarchical user enrollment using Oracle Common Directory Access (CDA) and the use of sub-documents.

For more information about Training, please see [Appendix A: Components of EAR and WAR Samples](#).

For more information about hierarchical enrollment and CDA, see the *SDK Guide for Oracle Siebel eStatement Manager*.

umfsample

The **ear-umfsample.ear** J2EE application implements a non-directory enrollment model that customizes the Oracle user management framework as an interface to enrollment information already stored in a separate repository.

umfsample does *not* ship with eStatement Manager, but is installed separately with the SDK. For details about working with **umfsample**, see the *SDK Guide for Oracle Siebel eStatement Manager*.

Sample Datasets for eaSuite

The samples directory also contains sample data and design files (DDF, ALF, and HTML templates) for use with sample Web applications.

Datasets are optimized to implement features of sample Web applications. The following table shows which datasets to use with each sample Web application.

Dataset/Web App	Sample	Training	umfsample
NatlWireless	X	X	X
NW_LDDetail	X	X	

About Jobs, Views, and Version Sets

Most features of eStatement Manager require that you use the Oracle Command Center to publish a specific type of *view*, sometimes with particular parameters and settings. A *view* is a set of design files that result in a particular presentation of statement data. The view files enable a user to dynamically display formatted statements live on the Web, receive email notifying them that an online statement is available, or to view account data in a static format online.

A feature may also depend on a particular sequence of *jobs*; for example, you must have loaded detail data into the database in order to dispute and annotate line items.

When creating and configuring your eStatement Manager application, you *publish* the files that make up each dynamic Web view in your application. Publishing adds a timestamp to the set of view files. A

Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager
Version 4.7

version set is a dated set of design files for a dynamic Web view. Publishing a view is also called creating a version set.

TIP: If you cannot see a particular feature in one of the samples, check to make sure that you have published the necessary views and run the required jobs in the correct sequence.

For more information about Jobs, Views, and Version Sets, see the *Administration Guide for Oracle Siebel eStatement Manager*.

3

Deploying and Using Sample

About Sample

Sample is a sample J2EE application that eStatement Manager provides as part of its software distribution. You can use it as a framework for developing a custom EJB application, as it contains all the Java Server Pages (JSPs), HTML, image files, scripts, and templates you need to get started. Sample deploys as *ear-sample.ear*.

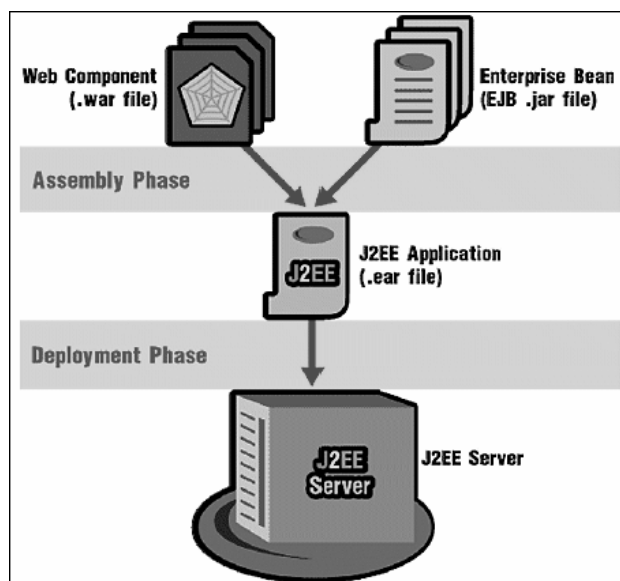
You can use Sample to view the sample NatlWireless and NW_LDDetail applications provided with eStatement Manager. You can use the data and design files in these sample applications to become familiar with eStatement Manager by creating sample billing applications and jobs, publishing data and design files in the form of *version sets*, and scheduling the jobs to run in the Command Center, the administrative 'hub' for the eStatement Manager production environment.

The following steps describe how to deploy and use Sample to view the sample eStatement Manager application called **NatlWireless**. NatlWireless is a set of example design and data files that demonstrate the features of an eStatement Manager presentment application.

You must set up NatlWireless in the eStatement Manager Command Center (production environment), then enroll and log in to Sample to view sample bills.

Deploying the Sample J2EE Web Application

When you install eStatement Manager, you will find the Enterprise Application Archive (EAR file) for eStatement Sample in the `<EDX_HOME>/samples/sample/j2eeapps` directory of your eStatement Manager installation. You deploy this application using your application server's administrative console.



In general, deploying a Web application involves three distinct phases:

- 1 Component creation, typically done by application developers.
- 2 Application assembly, typically done by application developers (although they may not have participated in the 'component creation' phase).
- 3 Application deployment, typically done by both application developers and system administrators.

During development and testing, it is common for Web developers to deploy their own applications. However, when the application has been assembled and is ready for production, a system administrator will most likely deploy it.

Deployment Using WebLogic

This section provides an example of how to deploy Sample using the WebLogic application server.

CAUTION: The specific steps for your application server may differ from the ones described below. You should consult your System Administrator and application server documentation for complete details of how to deploy a J2EE application based on your system's configuration.

The steps below summarize how to deploy using WebLogic Server 9.2. Please consult your BEA WebLogic documentation for specific instructions on this task.

Note that WebLogic Server will not deploy an application that has errors in its deployment descriptor. Previous versions of WebLogic Server would deploy an application that had errors in its deployment descriptor.

To deploy a Web application to your WebLogic application server:

- 1 Start WebLogic Server, if it isn't already running.
- 2 Open a Web browser and enter a URL with the syntax:

<http://host:port/console>

For example:

http://audi:7001/console

- 3 The Enter Network Password dialog appears. (Note that the dialog might take a few seconds to display.)
- 4 Enter your WebLogic Server user name and password. The default user name is 'system' and the password is the one you specified during installation of WebLogic Server.
- 5 Click **OK**. The WebLogic Server Console main menu appears.
- 6 Select Deployments and click on install.
- 7 Browse to the location of your EAR file, select it and click on next.
- 8 Select **Install this deployment as an application** and click **Next**.
- 9 Leave the default settings in the "Optional Settings" page and click **Finish**.
- 10 After activating the changes make sure that you select the newly deployed ear in the "Summary of Deployments" page and start it.

Deployment Using WebSphere

This section describes how to deploy Sample using the WebSphere application server.

CAUTION: The specific steps for your application server may differ from the ones described below. You should consult your System Administrator and application server documentation for complete details of how to deploy a J2EE application based on your system's configuration.

To deploy a Web application to your WebSphere application server:

- 1 Navigate to the WebSphere 6.1 administration console in the web browser using the proper URL in the format: `http://[host]:[port]/ibm/console`. For example:
http://172.20.2.10:9060/ibm/console
Note that the administration port number is assigned automatically at installation and may be different for your installation. If you enable security, it prompts you for the user name and password you entered during installation.
- 2 Click **Application** and **Enterprise Applications**. Click **Install**.
- 3 If application is on your local machine, select the **Local file system** option. If not, select **Remote file system** and browse to select the sample ear file.
- 4 Accept the default setting for other screens. Save your deployment setting to master configuration. Then select the deployed application and click **Start** to start the application.

Deployment Using Oracle Application Server

This section describes how to deploy Sample along with any other applications on Oracle Application Server. Oracle Application Server employs the parent/child ear concept and requires you to deploy the applications at the correct level.

It is important to make the eStatement ear the child of all the other ears (reportSample, Training, Sample, and Payment, if you deploy those). You can deploy other ears anywhere above in a single hierarchy. You can deploy any number of ear files in to a single hierarchy.

You must deploy the top-most parent first. When you deploy the second ear file, you must select the first ear as the parent.

Similarly, when you deploy the third ear, specify the second ear as the parent. Continue this until you deploy eStatement:

```
Payment
→ Sample
  → Training
    → reportSample
      → eStatement
```

CAUTION: The specific steps for your application server may differ from the ones described below. You should consult your System Administrator and application server documentation for complete details of how to deploy a J2EE application based on your system's configuration.

To deploy a Web application to Oracle Application Server:

- 1 Open your browser and enter the proper URL in the format: `http://<hostname>.<domain>:<port number>/`. For example:

```
http://172.20.2.69:7777/
```

Note that the administration port number is assigned automatically at installation and may be different for your installation. When prompted for user name and password, enter correct values and click **Login**.

- 2 On the Application Server Control Console's Home page, click the **Members** table.

Start the HTTP Server and the OC4J components – Do this by selecting the check box next to each component and then clicking the **Start** button.

- 3 Create an OC4J instance. Switch to the `AS_HOME/bin` directory, where `AS_HOME` is your application server installed directory, for example, in Linux:

```
/opt/oracle/product/10.1.3/OracleAS_1/bin
```

- 4 Run the following command at the console:

```
./createinstance -instancename <Your OC4J instance name>
```

Where `<Your OC4J instance name>` is the OC4J instance name (for example, eStatement) and provide the password for the OC4J instance (required).

- 5 Deploy the EAR J2EE Application: click the OC4J instance (for example, OracleInfo) and click **Applications** and **Deploy**.
- 6 Select the type of Archive, Deployment Plan and Browse for file path, and click **Next**.
- 7 Provide the application same, parent application, select **Bind Web Module to Site**, and click **Next**.
- 8 Under Deployment Task, select **Configure Class Loading** and import your created Shared Library. Click **OK** then **Deploy**.

Setting up NatlWireless

- 1 Create a new application for NatlWireless in the eStatement Manager Command Center.
- 2 Create a new Indexer job, publishing the application's indexing DDF for the job to use, configure the four tasks that run sequentially as part of the Indexer job, and run the job. Then publish the NatlWireless application (dynamic HTML view) files designed to display the statement summary.
- 3 Create and configure a Detail Extractor job, publishing the DDF, database table XML file, and statement XSLT stylesheet view files designed for the Detail Extractor job, and run the job. Then publish the three dynamic XML Query files (views) designed to display the extracted NatlWireless data and demonstrate the disputes and annotations features.

To create a new eStatement Manager application for NatlWireless:

- 1 Start your application server and the Scheduler, if not already running.
- 2 Open a Web browser and enter the URL to the eStatement Manager Command Center, for example:

`http://<HOST>:<PORT>/eBilling`
- 3 Create a new application for NatlWireless. Click **Create New Application** at the Main Console. The Create New Application screen appears.
- 4 Enter **NatlWireless** as the application name.
- 5 Enter the JNDI name of the datasource EJB as follows:

`edx/Sample/ejb/EdocsDataSource`
- 6 Click **Create Application and Continue**. eStatement Manager displays the Create New Job screen.

To create and configure an Indexer job:

- 1 The Create New Job screen appears after you create a new application.
- 2 Enter **Indexer** for the Indexer job name, and select the **Indexer** job type from the drop-down menu. Click **Launch Publisher** to publish the indexing DDF for NatlWireless. Click **Create**. The Publisher displays the Select a Version Set Type screen.

- 3 Under Batch Jobs, next to Indexer, click 0 (Number of Auxiliary files). The Publisher displays the Create a Version Set for Indexer screen.
- 4 Select **NatlWireless** from the list of application names, and browse to the \$EDX_HOME/samples/NatlWireless directory and select Indexerjob\NatlWirelessIndexer.DDF file for the Indexer job.
- 5 Click **Submit**. The Publisher displays the Submission screen with details about the DDF file. Close the Publisher window.
- 6 At the Create New Job screen in Command Center, click **Configure Job and Continue**. eStatement Manager displays the job configuration screen. For each task, specify the configuration parameters listed below:

Task 1: Scanner Task Configuration	
Input File Path	Use the default (EDX_HOME/Input/NatlWireless).
Input File Name	Specify NatlWireless.txt .
Output File Path	Use the default (EDX_HOME/Data/NatlWireless).
Task 2: Indexer Task Configuration	
DDF Path	(Not editable.)
Task 3: IXLoader Task Configuration	
Load Method	Use the default (Direct)
Task 4: AutoIndexVolAccept Task Configuration	
Action on Index Volume	Use the default (Auto Accept).

- 7 When finished entering the configuration parameters, click Submit Changes and Schedule. eStatement Manager asks "OK to submit this configuration?" Click OK. eStatement Manager submits the job configuration parameters and displays the Schedule screen.
- 8 In the left pane, click **Main Console**. On the Main Console's left pane, click **Publisher**, and then click **Create**. The Publisher displays the Select a Version Set Type screen.
- 9 Next to HTML under Dynamic Web Views, click 0. The Publisher displays the Create a Version Set for HTML screen.
- 10 Select **NatlWireless** from the drop-down list of application names. Enter **HtmlDetail** for the view name. Browse to \$EDX_HOME/samples/NatlWireless and select the **NatlWireless.DDF**, **NatlWireless.ALF**, and **NatlWireless.HTM** design files. Then click **Submit**. The Publisher displays the Submission screen showing the files you published.
- 11 Close the Publisher.
- 12 Move the NatlWireless data file (NatlWireless.txt), which is located in \$EDX_HOME/samples/NatlWireless/datafile, to \$EDX_HOME/Input/NatlWireless. This is the same data file that you specified when you configured the job.

- 13** On the Main Console, click the **Run Now** button next to the NatlWireless Indexer job. Monitor the job's progress by clicking **Refresh** on the Main Console window. The Indexer job completes successfully when the job status on the Main Console changes to "Done."

To create and configure a Detail Extractor job:

- 1** On the Main Console, click the application name, **NatlWireless**, listed under Applications in the table.
- 2** Click Add New Job. eStatement Manager displays the Create New Job screen.
- 3** Enter a job name (the job name can be whatever you want it to be), and then select job type **Detail Extractor**.
- 4** Click Launch Publisher. eStatement Manager displays the Publisher screen. Click **Create**. The Publisher displays the Select a Version Set Type screen.
- 5** Under Batch Jobs, next to Detail Extractor, click 0 (Number of Auxiliary files). The Publisher displays the Create a Version Set for Detail Extractor screen.
- 6** Select the **NatlWireless** application from the drop-down list. Enter the view name **dtlextr** (this name is hard coded in several JSPs for detail, disputes, and annotations).
- 7** Browse to select **NatlWireless.DDF**. (The default location for this file is \$EDX_HOME/samples/NatlWireless/NatlWireless.DDF.)
- 8** Browse to select **summary_info.XML**, the database table XML view file created for this job. (The default location for this file is \$EDX_HOME/samples/NatlWireless/DetailExtractor.)
- 9** Browse to select **summary_info.XSL**, the statement XSLT stylesheet. The default location for this file is \$EDX_HOME/samples/NatlWireless/DetailExtractor.
- 10** Click **Submit** and close the Publisher.
- 11** On the Create New Job screen in the Command Center, click **Configure Job and Continue**. eStatement Manager displays the Detail Extractor job configuration screen.
- 12** Specify the configuration parameters (below) for each of the three tasks in the Detail Extractor job:

Task 1: IVNScanner Task Configuration	
Field	What to enter/select
Index Volume Status	Choose the default, Accepted .
Scan Starting From (Number of Days)	Use the default (7).
Task 2: StatementsToIR Task Configuration	
Field	What to enter/select
View Name	dtlextr
Enroll Model	Leave blank.

Output File Path	Use the default (data output directory, specified in the Scanner task for the Indexer job).
Task 3: DXLoader Configuration	
Field	What to enter/select
Load Method	Use the default (Direct).

- 13 Click **Submit Changes and Schedule**. eStatement Manager asks "OK to submit this configuration?" Click **OK**. eStatement Manager submits the job configuration parameters and displays the Schedule screen.
- 14 On the Schedule screen, click **Run Now**.
- 15 On the left pane, click **Main Console**.
- 16 Publish the XMLQuery dynamic Web views that use the data extracted by the Detail Extractor; click **Publisher**.
- 17 Click **Create**. The Publisher displays the Select a Version Set Type screen. Under Dynamic Web Views, click the 0 next to the XML Query job type. The Publisher displays the Create a Version Set for XML Query screen.
- 18 Select the **NatlWireless** application. Enter **DetailQuery** as the view name, and browse `%EDX_HOME%\samples\NatlWireless\XMLQuery` to select the **detail_sql.xml** XML query file. (The DetailQuery view name is hard coded in your JSP HTML pages as the specific name the Web browser looks for in the code.) Click **Submit**. The Publisher displays the Submission screen.
- 19 Click **Create** and repeat the previous two steps twice to publish two additional XML Query views (both view names are hard coded in your JSP HTML pages as the specific names the Web browser looks for in the code):

View Name	File
DisputeQuery	<i>dispute_sql.xml</i>
AnnotationQuery	<i>annot_sql.xml</i>

- 20 Close the Publisher. You can proceed to use Sample to display the data.
- 21 On the Main Console, click the Run Now button next to the NatlWireless Detail Extractor job. Monitor the job's progress by clicking **Refresh** on the Main Console window. The Detail Extractor job completes successfully when the job status on the Main Console changes to "Done."

Viewing NatlWireless Statements in Sample

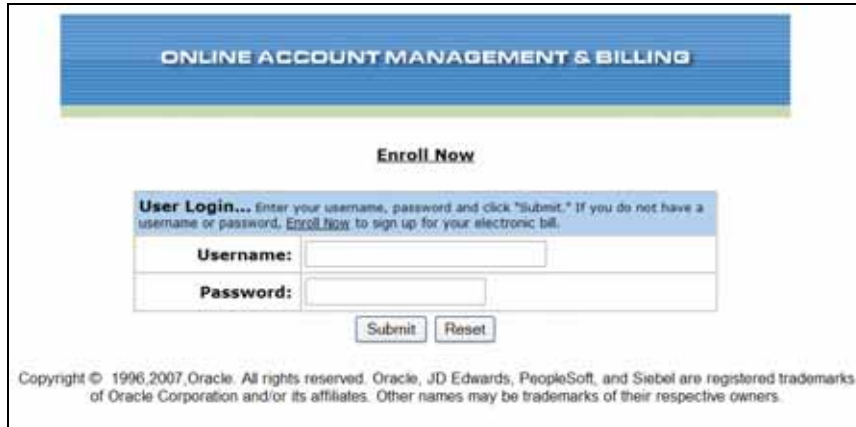
When the Indexer job completes successfully (status changes to "Done"), you are ready to view your online statements in Sample.

To use Sample to view NatlWireless statements:

- 1 Open a Web browser and access eStatement Sample, substituting your own server name (host) and port number:

```
http://<HOST>:<PORT>/Sample/User?app=UserMain&jsp=
/user/jsp/HistoryList.jsp&ddn=NatlWireless
```

The Sample User Login page appears:






- 2 Click the **Enroll Now** link. The sample enrollment page appears.



- 3 You can enter any user name and any password. However, you must enter a valid email address and a valid NatlWireless customer account number, such as one of the following: 0331734, 4191463, or 8611250. Use Reset to clear the text fields, if necessary.

- 4 Click **submit** to save the subscription information. eStatement Manager displays a message to let you know you have subscribed successfully. Click OK to display the User Login page.
- 5 Enter the username (Subscriber ID) and password (the same combination you entered during enrollment).
- 6 Click **submit**. The sample statement summary page for the account appears. (Note that you must have ePayment Manager installed to view the payment screens and functionality.)

Account History...Click icons to view account detail.				
View	Manage Statement	Z_DOC_DATE	AmtDue	DueDate
		12-04-2002	224.73	04/19/01 TO

- 7 To view the Account Summary page, click the View icon .





Account No	0001734	BILLS BICYCLES	Due	04/19/01
		44 HOLLY ST		
Statement	MARCH 28, 2001	WRENTHAM MA 02097	Amount Due	\$224.73

THANK YOU! FOR YOUR PATRONAGE.


ACCOUNT SUMMARY





























PREVIOUS BALANCE	285.12
LESS PAYMENTS APPLIED THROUGH 03/04/01	158.37 CR
MISCELLANEOUS CREDIT	19.19 CR
BEGINNING BALANCE	107.56
CURRENT USAGE	.07
PRODUCT MONTHLY FEES	4.95
CORPORATE CONNECTIONS WAIVER	4.95 CR
LIFELINE ASST/TELE RELAY	0.00
FEDERAL ACCESS CHARGE	8.62
LOCAL USAGE CHARGE	13.68
LOCAL SERVICE CHARGE	83.75
FEDERAL TAXES - LOCAL SERVICE	2.62
STATE TAXES - LOCAL SERVICE	2.15




A Slice with
Luxury Couch


Click here

To view the Manage Statement page, click the Manage Statement icon .

Comments	Summary Info Description	Summary Info Amount	Dispute
	PREVIOUS BALANCE	285.12	
	LESS PAYMENTS APPLIED THROUGH 03/24/01	158.37	
	MISCELLANEOUS CREDIT	19.19	
	BEGINNING BALANCE	107.56	
	CURRENT USAGE	.07	
	PRODUCT MONTHLY FEES	4.95	
	CORPORATE CONNECTIONS WAIVER	4.95	
	LIFELINE ASST/TELE RELAY	0.80	
	FEDERAL ACCESS CHARGE	8.62	
	LOCAL USAGE CHARGE	13.68	
	LOCAL SERVICE CHARGE	83.75	
	FEDERAL TAXES - LOCAL SERVICE	2.62	
	STATE TAXES - LOCAL SERVICE	2.15	
	LATE FEE	1.61	

- 8 On the Manage Statement page, click the Comments icon  next to an item to display the Add Note page. Here you can add comments (annotations) regarding that item:

Add note:	
Description:	CURRENT USAGE
Category:	Business ▼
Comment:	<div></div>
<div>Submit</div> <div>Reset</div>	

On the Manage Statement page, click the Dispute icon  to display the Dispute Your Statement page. This page lets you dispute the item:

Dispute your statement:	
Disputed Item:	Summary Info Amount
Current Amount:	4.95
Adjusted Amount:	<input type="text"/>
Comments:	<div><div></div><div></div></div>
<div><div>Submit</div><div>Reset</div></div>	

4

Renaming Sample to a New J2EE Application

About Application Contexts

To rename your Web application, change its *context name* and modify the *deployment descriptors* for each main component. This is the first step toward customization of any sample Web application.

The following chapter, [Creating Your Own Custom Web Application for eStatement Manager](#), gives an overview of some common customization choices and directs you to the appropriate SDK module for working with those components. Most customization involves the WAR file only, though if you are customizing enrollment, you may be working with an enrollment EJB, for example `ejb-enrollment-cda.jar`. Other EAR file EJBs should not be customized beyond modifying their deployment descriptors as shown below.

This chapter describes how to change the *context root* of Sample and update the *JNDI name* of each deployment descriptor to point to the new context root.

About the Context Root

A *context root* is a name that maps to the document root of a Web client. If your client's context root is `sample`, then the following request URL retrieves the file `index.html` from the document root `sample`:

```
http://<host>:<port>/sample/index.html
```

You do not need to change the context root, "sample" to the name of your new Web application.

For an introduction to context root and Web client configuration, see The J2EE Tutorial at <http://java.sun.com/j2ee/1.4/index.jsp>.

About JNDI Names

A *JNDI name* is a people-friendly name for an object in the Java Naming and Directory Interface API (JNDI). These names are bound to their objects by the J2EE server naming and directory service.

A *resource reference* is an element in a deployment descriptor that identifies the component's coded name for the resource. More specifically, the coded name references a connection factory for the resource. A *connection factory* is an object that produces connection objects that enable a J2EE component to access a resource.

The JNDI name of a resource and the name of the resource reference are *not the same*. You must change both the JNDI name and the resource reference for Sample to the name of your new Web application.

For an introduction to JNDI names, see The J2EE Tutorial at <http://java.sun.com/j2ee/1.4/index.jsp>.

About XML Deployment Descriptors

Each J2EE application, Web component, enterprise bean, and J2EE application client has a *deployment descriptor*. A deployment descriptor is an XML document that describes a component's deployment settings. An enterprise bean deployment descriptor, for example, declares transaction attributes and security authorizations for the bean. Deployment descriptors are located in the META-INF directory of each component.

TIP: Deployment descriptors for WebLogic and Oracle Application Server have the extension **.xml**. Descriptors for WebSphere have the extension **.xmi**.

For the XML DTDs for EAR, WAR, and EJB files in the J2EE specification, see <http://java.sun.com/j2ee/1.4/index.jsp>.

Define Your New Context Name

Define a *context name* for your new Web application. This name will be the Web context for all URL references; a prefix to the JNDI name references in all deployment descriptors and part of the name of the WAR and EAR files. Be consistent: use the same context name everywhere.

For the Sample EAR file, the context name is **sample**. This example uses **Newapp** as its context name, but you should use your own descriptive name.

Write your context name here:

Edit EAR File Deployment Descriptors

Extract the Sample EAR File

Before you can modify Sample, you must extract the contents of the enterprise archive to a working directory. This allows you to view and change individual components of the archive, in this example the context root and JNDI names only. Once you have made your changes, you must repackage the archive and redeploy it to your application server for the change to take effect.

To extract ear-sample.ear to a working directory:

- 1 Create a working directory on your machine, for example:

```
<EDX_HOME>/ear-newapp
```

- 2 Copy the file **ear-sample.ear**

```
From <EDX_HOME>/j2eeapps
```

```
To <EDX_HOME>/ear-newapp
```

- 3 Extract the files from the EAR file, for example:

```
jar xvf ear-sample.ear
```

Windows users can extract the contents of an EAR file with WinZip.

Once you have extracted the EAR file to your working directory, delete the **ear-sample.ear** file from that directory, or move it to a backup location. DO NOT delete the version in the j2eeapps directory!

TIP: The examples described in this guide presume you are deploying to a production environment, so you should clean up your working directory after each step before proceeding.

After the extraction, your working directory will contain the files listed in [Components of ear-sample.ear](#) and shown in the following diagram.

Architecture of Sample EAR

For a complete listing and definition of components in **ear-sample.ear**, please see [Appendix A: Component Lists for Sample Web Applications](#).

To rename your Web application, you must change *only the deployment descriptors* for the EAR file, the WAR file, and each EJB in the EAR file. Deployment descriptors are located in the META-INF directory of EAR and JAR files, and the WEB-INF directory of WAR files. The following sections describe each task in detail.

Edit application.xml

The **application.xml** deployment descriptor defines runtime properties of the J2EE application EAR file. Deployment descriptors are located in the WEB-INF directory of the EAR file. The following diagram illustrates the XML elements defining runtime properties of Sample.

TIP: XML illustrations in this SDK were created with Altova XML Spy. For information about XML Spy, see <http://www.altova.com>.

XML			
application			
xmlns	http://java.sun.com/xml/ns/j2ee		
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance		
xsi:schemaLoca...	http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/application_1_4.xsd		
version	1.4		
description	Copyright (c) 1996, 2006 Oracle. All rights reserved.		
display-name	sample		
module (11)			
1	web	ejb	
2	web-uri	war-sample.war	
3	context-root	Sample	
4		ejb-application.jar	
5		ejb-enrollment-cda.jar	
6		ejb-fileserver.jar	
7		ejb-ccmerger.jar	
8		ejb-xsltmerger.jar	
9		ejb-querymerger.jar	
10		ejb-annotation.jar	
11		ejb-dispute.jar	
		ejb-session.jar	
		ejb-versioning.jar	

From this example, we can see that the name of the J2EE application appears in three XML elements: the **display-name** of the application and both the **web-uri** and **context root** of its Web application WAR file.

To rename Sample in application.xml, search for all occurrences of **Sample** or **sample** (note case-sensitivity) and replace with **Newapp** or **newapp**.

The next step is to rename individual deployment descriptors inside each WAR and EJB component of the EAR file. Once you have finished renaming these components, you must [Repackage and Deploy the EAR File](#).

Edit EJB Deployment Descriptors

About Sample EJBs

As we can see in **application.xml**, Sample contains thirteen EJB JAR files. For a definition of each JAR file and which SDK features use it, see [Appendix A: Components of EAR and WAR Samples](#). To rename your application, you must edit deployment descriptors for *each* of these JAR files.

Each EJB contains three deployment descriptors that contain structural and application assembly information for each enterprise bean. Two of these, **ejb-jar.xml** (for the EJB itself) and **sun-ejb-jar.xml** (for the J2EE specification), are the same for all platforms. The application server descriptor for WebLogic is **weblogic-ejb-jar.xml**; for WebSphere, **ibm-ejb-jar-bnd.xmi** and for Oracle Application Server orion-ejb.jar.xml

The following sections describe how to edit **ejb-jar.xml** and either **weblogic-ejb-jar.xml** or **ibm-ejb-jar-bnd.xmi** or **orion-ejb.jar.xml**. You need not modify **sun-ejb-jar.xml**.

CAUTION: EJB files contain class files required by eStatement Manager for each J2EE application. DO NOT MODIFY these class files while you are renaming your application context. Edit only the XML files.

Extract Descriptors from the EJB JAR File

For each EJB in Sample, you should follow these steps:

- 1 Extract **ejb-jar.xml** from the JAR file.
- 2 Extract **weblogic-ejb-jar.xml** (WebLogic), **ibm-ejb-jar-bnd.xmi** (WebSphere) or **orion-ejb.jar.xml** (Oracle Application Server) from the JAR file.
- 3 Change all occurrences of **Sample** to **Newapp**, and **sample** to **newapp** (note case-sensitivity) in both files.
- 4 Repackage the JAR file with its two new descriptors.
- 5 Clean up your working directory before proceeding.

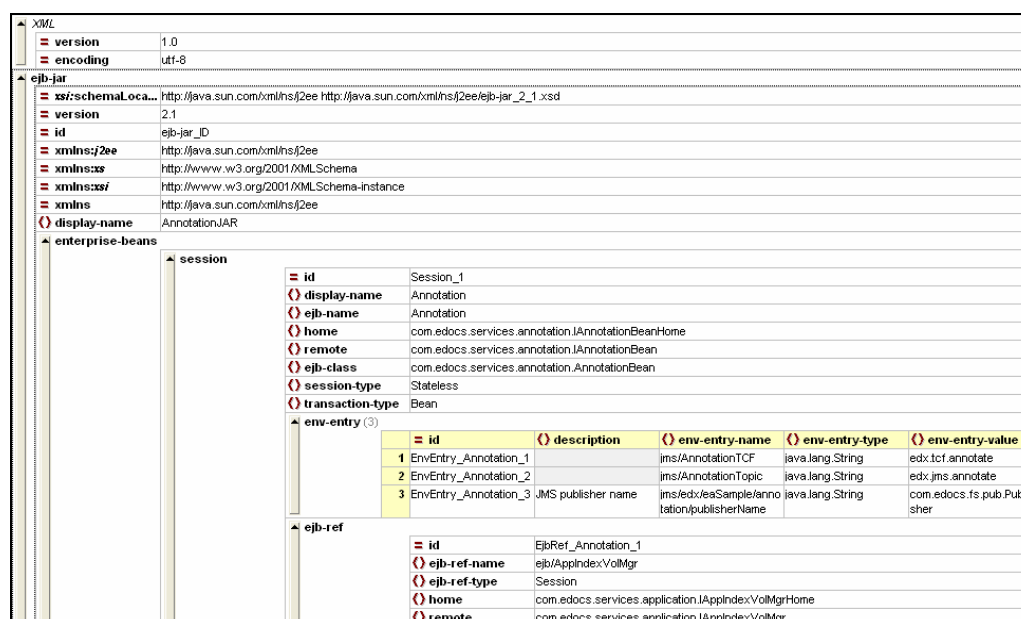
The following sections give a detailed example of each step.

TIP: As long as Sample is deployed on your application server, you may find that a new J2EE application works fine without renaming these files. However, to make your code more robust and

prevent errors (such as your session unexpectedly timing out), use the best practice of mapping all application server deployment descriptors to the JNDI name of your Web application.

Edit ejb-jar.xml

The `ejb-jar.xml` deployment descriptor defines runtime properties of the enterprise bean. Deployment descriptors are located in the WEB-INF directory of the EJB. The following diagram illustrates the XML elements defining runtime properties of `ejb-annotation.jar` in Sample.



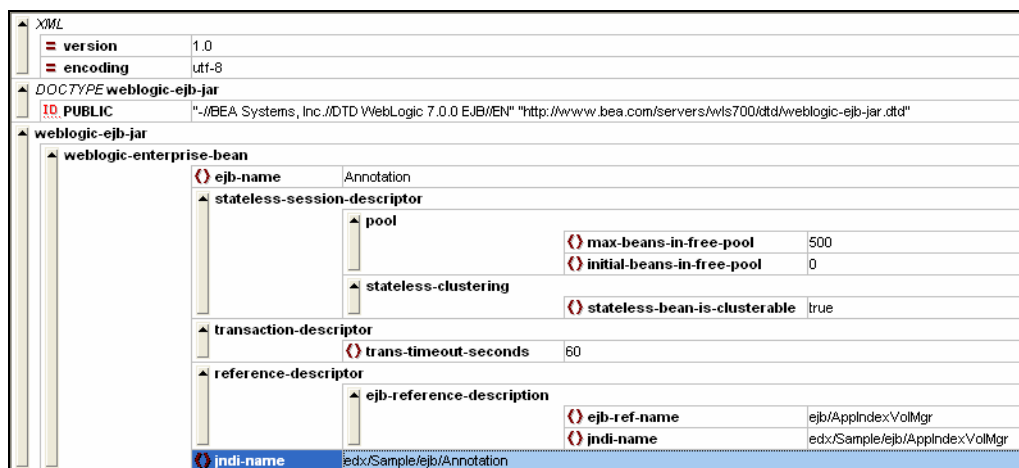
From this example, we can see that this `ejb-jar` deployment descriptor refers to a JAR file for a stateless session bean. This JAR is named **AnnotationJAR**, and its session properties define its home, remote, class, session, and reference types. These properties should not be modified.

This bean is mapped to the JMS Publisher name of its Web application in the third annotation entry of the **env-entry** property. The JMS publisher name contains the JNDI name of the bean itself, in this example `jms/edx/Sample/annotation/publisherName`.

You need to change this JNDI mapping of **Sample** to **newapp**, or the name of your own Web application. *You should make this change for every EJB in the Web application, in both the EAR and WAR files.*

Edit weblogic-ejb-jar.xml (WebLogic)

The `weblogic-ejb-jar.xml` deployment descriptor defines runtime properties of the enterprise bean for the BEA WebLogic application server. Deployment descriptors are located in the WEB-INF directory of the EJB. The following diagram illustrates the XML elements defining WebLogic properties of `ejb-annotation.jar` in Sample.



From this example, we can see that this `weblogic-ejb-jar` deployment descriptor refers to a WebLogic enterprise bean. This bean is named `Annotation`, and it is a stateless session bean that allows up to 500 instances in the free pool and may be clustered across multiple application servers. These properties should not be modified.

For more information about XML elements in `weblogic-ejb-jar`, see <http://edocs.bea.com/wls/docs92/ejb/>.

This bean is mapped to the JNDI name of its Web application in two places:

- The JNDI name of the bean itself, in this example `edx/Sample/ejb/Annotation`
- The JNDI name of its reference descriptor, `edx/Sample/ejb/AppIndexVolMgr`

To rename `Sample`, change *both* these JNDI mappings of `Sample` to `newapp` or the name of your own Web application.

Edit `ibm-ejb-jar-bnd.xmi` (WebSphere)

The `ibm-ejb-jar-bnd.xmi` deployment descriptor defines runtime properties of the enterprise bean for the IBM WebSphere application server. Deployment descriptors are located in the `WEB-INF` directory of the EJB. The following diagram illustrates the XML elements defining WebSphere properties of `ejb-annotation.jar` in `Sample`.

ejbBnd:EJBJarBinding	
xmi:version	2.0
xmi:id	ejb-jar_ID_Bnd
xmlns:ejb	ejb.xml
xmlns:common...	commonbnd.xml
xmlns:common	common.xml
xmlns:ejbBnd	ejbBnd.xml
xmlns:xmi	http://www.omg.org/XML
xmlns:xmins	xmins
ejbJar	href META-INF/ejb-jar.xml#ejb-jar_ID
ejbBindings	xmi:id Session_1_Bnd
	jndiName edx/Sample/ejb/Annotation
	ejbRefBindings
	xmi:id EjbRef_Annotation_1_Bnd
	jndiName edx/Sample/ejb/AppIndexVolMgr
	bindingEjbRef
	href META-INF/ejb-jar.xml#EjbRef_Annotation_1
	enterpriseBean
	xmi:type ejb:Session
	href META-INF/ejb-jar.xml#Session_1

From this example, we can see that this `ibm-ejb-jar-bnd.xml` deployment descriptor refers to a WebSphere enterprise bean (by noting that its root element is **EJBJarBinding**). WebSphere defines fewer properties in its EJB descriptor, and apart from the JNDI name, these properties should not be modified.

For more information about EJB bindings for WebSphere, see <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>.

This bean is mapped to the JNDI name of its Web application in two places:

- The JNDI name of the bean itself, in this example `edx/Sample/ejb/Annotation`
- The JNDI name of its EJB reference bindings, `edx/Sample/ejb/AppIndexVolMgr`

You need to change *both* these JNDI mappings of **sample** to **newapp**, or the name of your own Web application.

Repackage the EJB Jar File

Once you have modified both EJB descriptors, repackage the JAR file using the following command:

```
jar cvf ejb-application.jar com META-INF/weblogic-ejb-jar.xml META-INF/ejb-jar.xml _WL_GENERATED META-INF/sun-j2ee-ri.xml
```

Clean up your working directory before proceeding.

CAUTION: Windows users should *not* use WinZip to repackage components, as this can cause your application to deploy incorrectly.

Repeat for Each EJB in the EAR and WAR Files

Remember to rename EJB descriptors for every EJB in the Web application, in both the EAR and WAR files. You may want to use [Appendix A: Components of EAR and WAR Samples](#) as a checklist.

Edit WAR File Deployment Descriptors

Extract the Sample WAR File

The `war-sample.war` file contains example Web application components that you can customize or extend. The following chapter, [Creating Your Own Custom Web Application for eStatement Manager](#), gives an overview of some common customization choices and directs you to the appropriate SDK module for working with those components. Most customization involves individual WAR file components, primarily JSPs and HTML files. This section addresses only customizing the deployment descriptors to the new context and JNDI name.

Extract the contents of the WAR file to a temporary directory using the following command:

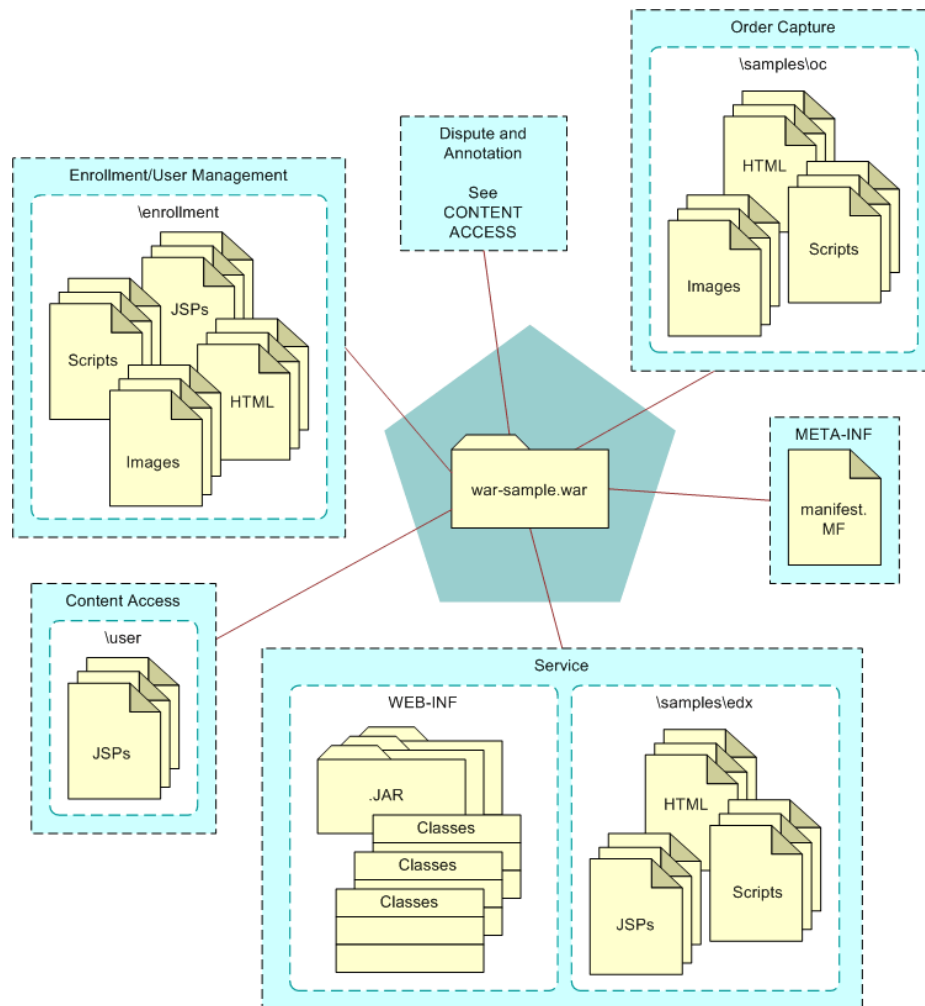
```
mkdir temp
jar xvf war-sample.war
```

Windows users can extract the contents of an EAR file with WinZip.

For a complete listing of components in each directory of `war-sample.war`, see [Appendix A: Component Lists for Sample Web Applications](#).

Architecture of the WAR File

After the extraction, your working directory contains the files listed in Components of `war-sample.war` and shown in the following diagram.



These include:

- *deployment descriptors* in the [WEB-INF](#) directory. You must customize each of these deployment descriptors as described in the next section.
- JSPs, JavaScripts, and HTML pages for *user management*, in the [enrollment](#) directory. To customize these files, see the *SDK Guide for Oracle Siebel eStatement Manager*.
- JSPs, JavaScripts, and HTML pages for *content access* and *line item dispute and annotation*, in the [user/jsp](#) directory. To customize these files, see *SDK Guide for Oracle Siebel eStatement Manager*.
- *Class files* and executable JAR files that reference servlets and EJBs, in the [WEB-INF](#) directory. You must customize the deployment descriptors for each of the JAR files.

Edit web.xml

The **web.xml** deployment descriptor is a complex file that defines the properties of your Web application WAR file.

For instructions on writing the **web.xml** deployment descriptor for BEA WebLogic, see <http://edocs.bea.com/wls/docs92/webapp/>.

For more information about **web.xml** elements for WebSphere, go to <http://www.ibm.com>.

To rename Sample in **web.xml**, search for all occurrences of **Sample** or **sample** (note case-sensitivity) and replace with **Newapp** or **newapp**. This changes the elements display name, servlet account name, and some EJB references. For definitions of **web.xml** elements for WebLogic, see http://edocs.bea.com/wls/docs92/webapp/web_xml.html.

The following illustration shows the ID and servlet properties for Sample. Note that this illustrates only part of **web.xml** for Sample.

XML	version	1.0
	encoding	utf-8
web-app	xmlns:schema...	http://java.sun.com/xmlns/j2ee http://java.sun.com/xmlns/j2ee/web-app_2_4.xsd
	version	2.4
	xmlns:j2ee	http://java.sun.com/xmlns/j2ee
	xmlns:xsi	http://www.w3.org/2001/XMLSchema
	xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
	xmlns	http://java.sun.com/xmlns/j2ee
	display-name	Sample
	distributable	
servlet (6)	display-name	
	1	configuration-init
	2	UserService
	3	UserDispute
	4	UserAnnotation
	5	UserEnrollmentServlet
	6	ChartServlet
	servlet-name	
	1	configuration-init
	2	UserService
	3	UserDispute
	4	UserAnnotation
	5	UserEnrollmentServlet
	6	ChartServlet
	servlet-class	
	1	com.edocs.fs.logging.config.api.SConfiguration
	2	com.edocs.app.AppServlet
	3	com.edocs.app.AppServlet
	4	com.edocs.app.AppServlet
	5	com.edocs.app.AppServlet
	6	com.edocs.app.AppServlet
	init-param	
	1	ServletRoot
	2	ErrorPage
	3	LoginRoot
	4	LoginPage
	5	Account.name
	param-name	
	1	ServletRoot
	2	ErrorPage
	3	LoginRoot
	4	LoginPage
	5	Account.name
	param-value	
	1	com.edocs.app.user
	2	/enrollment/jsp/UserErrorMsg.jsp
	3	com.edocs.app.enrollment
	4	/enrollment/jsp/UserLogin.jsp
	5	edx/Sample/ejb/CDAAccount
	load-on-startup	
	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	servlet-mapping (5)	
	session-config	
	welcome-file-list	
	jsp-config	
	resource-ref id=WebRef_1	
	env-entry	
	ejb-ref (14)	

You must change the *display name* from **Sample** to **Newapp** or the name of your custom Web application.

Sample uses five *servlets* as illustrated. You also need to change the **Account.name** parameter for each servlet. Here is an XML example of the Account name parameter for the **UserService**:

```
<init-param>
<param-name>Account.name</param-name>
<param-value>edx/Sample/ejb/CDAAccount</param-value>
</init-param>
```

For *each* servlet, you must change the **Account.name** value (shown in bold) from **edx/Sample/ejb/CDAAccount** to **edx/newapp/ejb/CDAAccount**. For more information on customizing account names, see the *SDK Guide for Oracle Siebel eStatement Manager*.

The Sample WAR file also references 23 EJBs in various components of the EAR file. Each reference defines the type of bean and the location of its home and remote interface classes. This illustration shows the EJB references for Sample in **web.xml**.

ejb-ref (14)					
	id	ejb-ref-name	ejb-ref-type	home	remote
1	EjbRef_1	ejb/AppMgr	Session	com.edocs.services.application.IAppMgrHome	com.edocs.services.application.IAppMgr
2	EjbRef_2	ejb/AppIndexVolMgr	Session	com.edocs.services.application.IAppIndexVolMgrHome	com.edocs.services.application.IAppIndexVolMgr
3	EjbRef_3	ejb/AppVolMgr	Session	com.edocs.services.application.IAppVolMgrHome	com.edocs.services.application.IAppVolMgr
4	EjbRef_4	ejb/FileManager	Session	com.edocs.services.fileserver.IFileManagerHome	com.edocs.services.fileserver.IFileManager
5	EjbRef_5	ejb/CSVMerger	Session	com.edocs.services.merger.IMergerHome	com.edocs.services.merger.IMerger
6	EjbRef_6	ejb/HTMLMerger	Session	com.edocs.services.merger.IMergerHome	com.edocs.services.merger.IMerger
7	EjbRef_7	ejb/XMLMerger	Session	com.edocs.services.merger.IMergerHome	com.edocs.services.merger.IMerger
8	EjbRef_8	ejb/XSLTMerger	Session	com.edocs.services.merger.IMergerHome	com.edocs.services.merger.IMerger
9	EjbRef_9	ejb/XMLQueryMerger	Session	com.edocs.services.merger.IMergerHome	com.edocs.services.merger.IMerger
10	EjbRef_10	ejb/Session	Session	com.edocs.services.session.ISessionHome	com.edocs.services.session.ISession
11	EjbRef_11	ejb/VersionManager	Session	com.edocs.services.versioning.IVersionManagerHome	com.edocs.services.versioning.IVersionManager
12	EjbRef_12	ejb/VersionSetReader	Session	com.edocs.services.versioning.IVersionSetReaderHome	com.edocs.services.versioning.IVersionSetReader
13	EjbRef_13	ejb/Annotation	Session	com.edocs.services.annotation.IAnnotationBeanHome	com.edocs.services.annotation.IAnnotationBean
14	EjbRef_14	ejb/Dispute	Session	com.edocs.services.dispute.IDisputeBeanHome	com.edocs.services.dispute.IDisputeBean

You do not need to change these EJB references at this stage, but you will need to update them if you add custom components to your WAR file. Note that these references *must* map to the EJB references in **weblogic.xml** or **ibm-ejb-jar-bnd.xmi** as shown in the following sections.

Edit weblogic.xml (WebLogic)

The **weblogic.xml** deployment descriptor binds each EJB in the EAR file to its JNDI name in the WAR file. The following illustration shows the EJB reference descriptions for Sample in WebLogic. Note that each maps to an EJB reference in **web.xml**, which in turn points back to the bean interface classes.

XML

DOCTYPE weblogic-web-app

ID PUBLIC

"/BEA Systems, Inc./DTD Web Application 7.0/EN"

"http://www.bea.com/servers/wls700/dtd/weblogic700-web-jar.dtd"

weblogic-web-app

reference-descriptor

resource-description

res-ref-name

jdbc/DataSource

jndi-name

edx.user.databasePool

ejb-reference-description (14)

	ejb-ref-name	jndi-name
1	ejb/AppMgr	edx/Sample/ejb/AppMgr
2	ejb/AppIndexVolMgr	edx/Sample/ejb/AppIndexVolMgr
3	ejb/AppVolMgr	edx/Sample/ejb/AppVolMgr
4	ejb/CSVMerger	edx/Sample/ejb/CCMerger
5	ejb/HTMLMerger	edx/Sample/ejb/CCMerger
6	ejb/XMLMerger	edx/Sample/ejb/CCMerger
7	ejb/XSLTMerger	edx/Sample/ejb/XSLTMerger
8	ejb/XMLQueryMerger	edx/Sample/ejb/XMLQueryMerger
9	ejb/Session	edx/Sample/ejb/Session
10	ejb/VersionSetReader	edx/Sample/ejb/VersionSetReader
11	ejb/VersionManager	edx/Sample/ejb/VersionManager
12	ejb/FileServer	edx/Sample/ejb/FileServer
13	ejb/Annotation	edx/Sample/ejb/Annotation
14	ejb/Dispute	edx/Sample/ejb/Dispute

To rename Sample in **weblogic.xml**, search for all occurrences of **Sample** or **sample** (note case-sensitivity) and replace with **Newapp** or **newapp**. This maps each EJB to its correct JNDI name.

Edit ibm-web-bnd.xmi (WebSphere)

The **ibm-web-bnd.xmi** deployment descriptor binds each EJB in the EAR file to its JNDI name in the WAR file. The following illustration shows the EJB reference descriptions for Sample in WebSphere. Note that each maps to an EJB reference in **web.xml**, which in turn points back to the bean interface classes.

webappbnd:WebAppBinding

xmi:version	2.0																																													
xmi:id	_Bnd																																													
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance																																													
xmlns:webapplication	webapplication.xml																																													
xmlns:commonbnd	commonbnd.xml																																													
xmlns:common	common.xml																																													
xmlns:webappbnd	webappbnd.xml																																													
xmlns:xmi	http://www.omg.org/XMI																																													
xmlns:xm:ns	xm:ns																																													
virtualHostName	xsi:nil=true xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance																																													
webapp	href=/WEB-INF/web.xml#																																													
ejbRefBindings (14)																																														
	<table><tr><th>xmi:id</th><th>jndiName</th><th>bindingEjbRef</th></tr><tr><td>1 EjbRef_1_Bnd></td><td>edx:Sample/ejb/AppMgr</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_1</td></tr><tr><td>2 EjbRef_2_Bnd></td><td>edx:Sample/ejb/AppIndexVolMgr</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_2</td></tr><tr><td>3 EjbRef_3_Bnd></td><td>edx:Sample/ejb/AppVolMgr</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_3</td></tr><tr><td>4 EjbRef_4_Bnd></td><td>edx:Sample/ejb/FileServer</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_4</td></tr><tr><td>5 EjbRef_5_Bnd></td><td>edx:Sample/ejb/CCMerger</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_5</td></tr><tr><td>6 EjbRef_6_Bnd></td><td>edx:Sample/ejb/CCMerger</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_6</td></tr><tr><td>7 EjbRef_7_Bnd></td><td>edx:Sample/ejb/CCMerger</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_7</td></tr><tr><td>8 EjbRef_8_Bnd></td><td>edx:Sample/ejb/XSLTMerger</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_8</td></tr><tr><td>9 EjbRef_9_Bnd></td><td>edx:Sample/ejb/XMLQueryMerger</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_9</td></tr><tr><td>10 EjbRef_10_Bnd></td><td>edx:Sample/ejb/Session</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_10</td></tr><tr><td>11 EjbRef_11_Bnd></td><td>edx:Sample/ejb/VersionManager</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_11</td></tr><tr><td>12 EjbRef_12_Bnd></td><td>edx:Sample/ejb/VersionSetReader</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_12</td></tr><tr><td>13 EjbRef_13_Bnd></td><td>edx:Sample/ejb/Annotation</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_13</td></tr><tr><td>14 EjbRef_14_Bnd></td><td>edx:Sample/ejb/Dispute</td><td>bindingEjbRef href=/WEB-INF/web.xml#EjbRef_14</td></tr></table>	xmi:id	jndiName	bindingEjbRef	1 EjbRef_1_Bnd>	edx:Sample/ejb/AppMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_1	2 EjbRef_2_Bnd>	edx:Sample/ejb/AppIndexVolMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_2	3 EjbRef_3_Bnd>	edx:Sample/ejb/AppVolMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_3	4 EjbRef_4_Bnd>	edx:Sample/ejb/FileServer	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_4	5 EjbRef_5_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_5	6 EjbRef_6_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_6	7 EjbRef_7_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_7	8 EjbRef_8_Bnd>	edx:Sample/ejb/XSLTMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_8	9 EjbRef_9_Bnd>	edx:Sample/ejb/XMLQueryMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_9	10 EjbRef_10_Bnd>	edx:Sample/ejb/Session	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_10	11 EjbRef_11_Bnd>	edx:Sample/ejb/VersionManager	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_11	12 EjbRef_12_Bnd>	edx:Sample/ejb/VersionSetReader	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_12	13 EjbRef_13_Bnd>	edx:Sample/ejb/Annotation	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_13	14 EjbRef_14_Bnd>	edx:Sample/ejb/Dispute	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_14
xmi:id	jndiName	bindingEjbRef																																												
1 EjbRef_1_Bnd>	edx:Sample/ejb/AppMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_1																																												
2 EjbRef_2_Bnd>	edx:Sample/ejb/AppIndexVolMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_2																																												
3 EjbRef_3_Bnd>	edx:Sample/ejb/AppVolMgr	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_3																																												
4 EjbRef_4_Bnd>	edx:Sample/ejb/FileServer	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_4																																												
5 EjbRef_5_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_5																																												
6 EjbRef_6_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_6																																												
7 EjbRef_7_Bnd>	edx:Sample/ejb/CCMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_7																																												
8 EjbRef_8_Bnd>	edx:Sample/ejb/XSLTMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_8																																												
9 EjbRef_9_Bnd>	edx:Sample/ejb/XMLQueryMerger	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_9																																												
10 EjbRef_10_Bnd>	edx:Sample/ejb/Session	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_10																																												
11 EjbRef_11_Bnd>	edx:Sample/ejb/VersionManager	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_11																																												
12 EjbRef_12_Bnd>	edx:Sample/ejb/VersionSetReader	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_12																																												
13 EjbRef_13_Bnd>	edx:Sample/ejb/Annotation	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_13																																												
14 EjbRef_14_Bnd>	edx:Sample/ejb/Dispute	bindingEjbRef href=/WEB-INF/web.xml#EjbRef_14																																												
resRefBindings	xmi:id=WebRef_1_Bnd jndiName=edx: user databasePool																																													

To rename Sample in **ibm-web-bnd.xml**, search for all occurrences of **Sample** or **sample** (note case-sensitivity) and replace with **Newapp** or **newapp**. This maps each EJB to its correct JNDI name.

Earlier versions of Sample contained hard-coded references to its context root on its welcome file, **user/jsp/index.jsp**. This JSP now requests the context root with the method call **String context = request.getContextPath()**. The only references to Sample in **index.jsp** are displayed as HTML text in the title and instructional URL, which is usually removed as part of customization. Oracle recommends calling **request.getContextPath()** as a best practice when customizing welcome or index pages.

Package the WAR File

When you have finished editing deployment descriptors for all WAR file components, repackage the WAR file using this command:

```
jar cvf war-newapp.war enrollment user WEB-INF
```

It is important to clean up your working directory after packaging and BEFORE extracting the next JAR file. If you skip cleanup, the next JAR files you edit may become corrupted with extraneous files.

CAUTION: Windows users should *not* use WinZip to repackage components, as this can cause your application to deploy incorrectly.

Repackage and Deploy the EAR File

Package the EAR file with a new name using the following command:

```
jar cvf ear-newapp.ear *
```

Deploy your custom application on your application server using the instructions in the *Installation Guide for Oracle Siebel eStatement Manager*.

To view your application, follow the instructions in the Installation and Configuration Guide for Oracle eStatement Manager for Sample, replacing **sample** in the URL with **newapp** or the context name of your new Web application.

TIP: Remember, you must index a dataset, publish a dynamic Web view, enroll a user, and log in to see your statement data online.

5

Creating a Custom Web Application for eStatement Manager

Define a Custom Enrollment Model

The steps to defining your custom enrollment model vary depending on which type you choose: CDA, LDAP, or non-directory access models. In any case, customizing your enrollment model affects the enrollment EJB you select, for example `ejb-enrollment-cda.jar` in Sample, and the enrollment information for the servlets in the WAR file.

For more information on choosing an enrollment model and customizing required files, see the *SDK Guide for Oracle Siebel eStatement Manager*.

Define Custom Servlets

Servlets in a Web application follow the architecture of the WAR file. If you create new class files for servlets, you should place them in a new subdirectory in the `WEB-INF/classes/com` directory.

For example, both the `UserServlet` and `UserEnrollment` servlet use the `AppServlet.class` file located in `WEB-INF/classes/com/edocs/app`. If you want to create a new `AppServlet.class` file, you should create it in a new subdirectory rather than replacing the existing file.

To create a custom servlet:

- 1 Write and compile your servlet class file, for example, `AppServletNew.class`.
- 2 Create a new directory in `WEB-INF/classes/com`, for example, `/newapp` and move your class file.
- 3 Modify the deployment descriptor files as described in preceding sections.

Customize the WAR File

The enrollment, samples, and user directories have the same basic subdirectory structure. Use this structure when creating new directories, for clarity.

- *help* — Any help files associated with the application.
- *html* — Static display HTML pages, such as error pages.
- *images* — Common GIF or JPEG files by the HTML pages or JSPs.
- *jsp* — Dynamic Web application pages that are the primary interface for your custom application.

- *scripts* — Any script files required for processing by a JSP. For example, the sorting feature of eStatement Manager requires the `script.js` file.

- *templates* — Any template files used for your application.

CAUTION: When creating or modifying Web application files, follow the eStatement Manager API Specification to reference eStatement Manager APIs correctly.

After completing your changes, see [Package the WAR File](#). Be sure to include any new WAR file directories in your `jar` command.

CAUTION: Do not include the JAR file *edx_client.jar* in the *WEB-INF/lib* directory of your customized WAR file. This file belongs only in the *lib* directory of the EAR file, and a copy inside the WAR file could cause your application to fail to deploy correctly.

You must then edit, package, and deploy to test your application.

Implement JSP Validation

About Validation of HTTP Requests

It is common practice to embed HTTP REQUEST parameters into the generated HTML page, for any number of reasons. If such a parameter is hacked to contain HTML content, for example to invoke a JavaScript function, the browser may end up executing malicious code within an authenticated session. This “cross-site scripting” can be very damaging.

For background and discussion, see the *CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests* at <http://www.cert.org/advisories/CA-2000-02.html>.

To limit this vulnerability, Oracle requires that all REQUEST parameters be checked to ensure that they contain no “active content” that might somehow trigger unwanted behavior when delivered back to an HTML browser.

eStatement Manager sample applications include the JSP code fragment in `Validation.jsp`, to be included in application-specific JSP pages. This fragment contains a **ValidatorBean** that validates the input and throws an exception in case of danger. Since this fragment needs to be customized per site or per application, some support is provided for helping determine the needed customizations.

Package `com.edocs.common.web.validation` Description

This API package implements the class **ValidatorBean** to support validation of input to JSP pages. Bean methods capture, set, validate and write the list of legal and illegal parameter names and values in a `ServletRequest` object.

You can use a standard server-side include to call this JSP fragment, as illustrated in these lines from the beginning of `Detail.jsp`:

```
// From Detail.jsp
<%@ include file="Validation.jsp" %>
```

About the ValidatorBean

The `ValidatorBean` is intended to be instantiated in a JSP page via the `<jsp:useBean>` tag, and the methods on the bean should validate all parameters for every REQUEST.

The list of illegal substrings is strongly related to the character set encoding. The sample provided assumes ISO-8859-1.

The list of illegal substrings is essentially a collection of the characters '`<`' and '`>`' in all the various encodings. There may well be an interaction with regex dynamic patterns.

Clearly, this discussion assumes use in a JSP context. For pure servlet applications, it should be possible to call the bean directly, though this has not been tested.

During development (especially when using the parameter name checking) you may want to set the scope the bean will page. For production this can be changed to application, which improves performance by caching the created and initialized bean.

In some limited cases it may be necessary to allow illegal substrings for certain parameters. This can be achieved by designating them as exempt, using the `setExemptParameterNames()` method.

Once a REQUEST has been checked, an attribute is set on the REQUEST object that alerts the validator that the object been checked, and should not be checked again. This prevents infinite recursion when the error page also performs parameter checking, since this recursion has been known to crash application servers.

CAUTION: Forwarding requests between pages and modifying the URL at the same time can introduce security holes, because the REQUEST is not checked again in the destination JSP.

To create and initialize a ValidatorBean:

```
<jsp:useBean id="validator"
class="com.edocs.common.web.validation.ValidatorBean"
scope="application">
<%
    // note that the entity-reference style encoding appears
    without the trailing semicolon
    //
    // the reason is that some browsers are too helpful and will
    guess that you
    // meant to include it. Rather than putting both &lt; and
    &lt; in the list,
    // we really need only the short one...
    //
    // if you use a different encoding, you will need to rethink
    this list.
    //
    String[] iso_8859_1_values =
    {"<", ">", "&lt;", "&gt;", "%3c", "%3e", "&#60", "&#62"};
    validator.setIllegalParameterSubStrings(iso_8859_1_values);
%>
</jsp:useBean>
```

In general, the only time you'll need to modify this is if you use a different encoding, which might cause the browser to treat more/different characters as "active" in the same way as '<' and '>'.

To invoke a ValidatorBean:

```
<%
    response.setContentType("text/html; charset=ISO-8859-1");
    validator.validateParameters(request);
%>
```

This method checks all REQUEST parameter names and values for illegal substrings and throws a `ValidationException` if any illegal substring is detected.

Using ValidatorBean for Parameter Name Checking

The `ValidatorBean` can also be used to validate the exact identities of parameter names. Using the `ValidatorBean` to check identities of parameters requires that you have an exact list of the parameters the application uses. Generating this list can be daunting, so the `ValidatorBean` implements some support for collecting this list. During `DEVELOPMENT`, a flag is set to capture parameter names and record them. For production, the flag must be cleared and the captured names put into the `Validation.jsp` page.

Because the penalty for failing a validation step is an exception, validation is typically disabled during development, and enabled only when the application is otherwise working. The task of discovering the list of valid parameter names can be eased somewhat by using Capture Mode.

In capture mode, the bean collects all request parameter names in an internal list, and prints them out when asked (via `writeParameterNames()` method). No checks are actually performed, and no exceptions thrown.

To enter Capture Mode, call `setCaptureMode()` with `true` as the argument. The validation methods will not throw an exception, but the `validateParameterNames()` method accumulates the names of all parameters it sees. These can be output at any time by calling `writeParameterNames()` method. The output can help create the complete list of legal names.

If some parameter is designated as *exempt*, its identity is still checked against the list of legal parameter names, but its *value* is not checked.

In validation mode, required for production, the bean actually performs the validations as required, and throws a `ValidationException` when some parameter name or value is invalid.

The list of legal parameter names in the following example is specific to an eStatement Manager sample application. In practice, the list of legal parameter names varies widely among applications, so this procedure must be configurable.

To validate parameter names:

```

<jsp:useBean id="validator"
class="com.edocs.common.web.validation.ValidatorBean"
scope="application">
<%
    // this is the list of all legal request parameter names
    String[] params = { "auth__userPassword", "docId", "ddn",
        "jsp", "app", "auth__uid", "APP_METHOD",
        "errforwardto", "EDOCSLOGIN", "edocs__re-login",
        "auth__dn", "Submit", "viewType", "viewName",};

    validator.setLegalParameterNames(params);

    String[] iso_8859_1_values =
    {"<", ">", "&lt;", "&gt;", "%3c", "%3e", "&#60", "&#62"};
    validator.setIllegalParameterSubStrings(iso_8859_1_values);

    // set this during development to accumulate parameter names
    validator.setCaptureMode(true);
%>
</jsp:useBean>

```

The following JSP fragment shows the invocation of the bean:

```

<%
    response.setContentType("text/html; charset=ISO-8859-1");

    // call to check that only parameters from the name list are
    present
    validator.validateParameterNames(request);

    // call to check for illegal substrings
    validator.validateParameters(request);

    // call DURING DEVELOPMENT to write out accumulated
    parameter names
    validator.writeParameterNames(out);
%>

```


6

Appendix A: Components of EAR and WAR Samples

Components of ear-sample.ear

The EAR file contains a customizable Web application archive (WAR file) and service EJBs to support eStatement Manager. EJB JAR files should NOT be modified or customized beyond modifying their deployment descriptors as described in this guide.

<root> directory of ear-sample.ear

Name	Purpose
<code>ejb-annotation.jar</code>	Service EJB for annotation of detail data. Used by <i>Line Item Dispute and Annotation</i> .
<code>ejb-application.jar</code>	Service EJB for management of DDNs. Used by all modules.
<code>ejb-ccmerger.jar</code>	Service EJB for composition of views. Used by all modules.
<code>ejb-dispute.jar</code>	Service EJB for line item dispute of detail data. Used by <i>Line Item Dispute and Annotation</i> .
<code>ejb-enrollment-cda.jar</code>	Java class and descriptor files for defining a User Management EJB with CDA. Used by <i>Implementing a User Management Framework</i> .
<code>ejb-fileserver.jar</code>	Service EJB for file transfer from Web to application tier. Used by all modules.
<code>ejb-querymerger.jar</code>	Service EJB for processing XML queries. Used by <i>Content Access</i> and <i>Line Item Dispute and Annotation</i> .
<code>ejb-session.jar</code>	Service EJB for application session management. Used by all modules.
<code>ejb-versioning.jar</code>	Service EJB for versioning of views. Used by all modules.
<code>ejb-xsltmerger.jar</code>	Service EJB for processing XSLT stylesheets. Used by <i>Content Access</i> and <i>Line Item Dispute and Annotation</i> .
<code>war-sample.war</code>	Customizable Web Application archive file containing JSPs, servlets, and descriptor files required for presentment. For more information, see Components of war-sample.war .

TIP: Note to customers migrating from previous versions: Sample no longer contains the service EJB `ejb-log.jar`. Logging in eStatement Manager 2.x is handled internally and no longer requires this Web application component.

/lib directory

Name	Purpose
<code>dom4j-1.6.1.jar</code>	Service JAR file containing Document Object Model (DOM) Java interface for dynamic access to HTML documents. Used by all modules. For more information, see http://www.w3.org/DOM/ .
<code>edx_client.jar</code>	Service EJB containing client APIs for access to other EJBs in the archive. Used by all modules. For more information, see the eStatement Manager SDK API Specification.
<code>edx_common.jar</code>	Service EJB containing Java classes common to all beans. Used by all modules. For more information, see the eStatement Manager SDK API Specification.
<code>edx_servlet.jar</code>	Service JAR file containing Java classes for Web APIs into eStatement Manager. Used by all modules. <i>Note:</i> This file is required in the <code>/lib</code> directory for each WAR file. It is also included at the EAR level for convenience when building a new Web application. Sample Web applications include <code>edx_servlet.jar</code> in both <code>*.ear/lib</code> and <code>*.ear/*.war/web-inf/lib</code> .
<code>xalan.jar</code>	Service JAR file containing Apache's Java implementation of the DOM API as an XSLT stylesheet processor. eStatement Manager uses xalan-Java-2.2 D12 as the fastest XSLT processor currently available. For more information, see http://xml.apache.org/
<code>xalanj1compat.jar*</code>	Service JAR file containing xalan-Java-1 XSLT processor, which is no longer supported by Apache but is required by the DOM4 API.
<code>antlr-2.7.6.jar</code>	ANTLR is a parser generator: a program that generates code to translate a specified input language into a nice, tidy data structure. This is a Spring dependency.

Name	Purpose
<code>aopalliance-1.0.jar</code>	Ensure interoperability between Java/J2EE AOP implementations. This is a Spring dependency.
<code>asm-1.5.3.jar</code>	Lightweight bytecode translator. This is a Spring dependency.
<code>asm-attrs-1.5.3.jar</code>	ASM library. Provides an implementation for various optional class, field, method and bytecode attributes. This is a Spring dependency.
<code>asm-util-1.5.3.jar</code>	ASM class visitors that can be useful for programming and debugging purposes. This is a Spring dependency.
<code>c3p0-0.9.0.jar</code>	c3p0 is an easy-to-use library for augmenting traditional (DriverManager-based) JDBC drivers with JNDI-bindable DataSources, including DataSources that implement Connection and Statement Pooling, as described by the jdbc3 spec and jdbc2 std extension. This is a Spring dependency.
<code>cglib-2.1_3.jar</code>	cglib is a powerful, high performance and quality Code Generation Library. It is used to extend JAVA classes and implements interfaces at runtime.
<code>com.infonyte.base.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>com.infonyte.collection.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>com.infonyte.index.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>com.infonyte.pdom.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>com.infonyte.pxslt.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>com.infonyte.xpath.jar</code>	The Infonyte product suite provides industrial strength storage, querying, and processing for native XML.
<code>commons-beanutils-1.7.jar</code>	The BeanUtils component provides easy-to-use wrappers for dynamic access to Java object properties.
<code>commons-collections-3.2.jar</code>	Powerful data structures that accelerate development of most significant Java applications
<code>commons-digester-1.7.jar</code>	Many Jakarta projects read XML configuration files to provide initialization of various Java objects within the system.

Name	Purpose
	There are several ways of doing this, and the <code>Digester</code> component was designed to provide a common implementation that can be used in many different projects.
<code>commons-lang-2.1.jar</code>	The standard Java libraries fail to provide enough methods for manipulation of its core classes. The <code>Lang</code> Component provides these extra methods.
<code>commons-logging-1.1.jar</code>	The Logging package is an ultra-thin bridge between different logging implementations. A library that uses the commons-logging API can be used with any logging implementation at runtime. Commons-logging comes with support for a number of popular logging implementations, and writing adapters for others is a reasonably simple task
<code>commons-pool-1.3.jar</code>	Pool provides an Object-pooling API, with three major aspects: A generic object pool interface that clients and implementors can use to provide easily interchangeable pooling implementations. A toolkit for creating modular object pools. Several <u>general purpose pool implementations</u>
<code>concurrent-1.3.3.jar</code>	This package provides standardized, efficient versions of utility classes commonly encountered in concurrent Java programming.
<code>edx_system.jar</code>	eStatement System library
<code>ehcache-1.1.jar</code>	Ehcache is a widely used java distributed cache for general purpose caching, J2EE and light-weight containers. It features memory and disk stores, replicate by copy and invalidate, listeners, a gzip caching servlet filter and much more. This is a <u>Spring dependency</u> .
<code>hibernate-3.1.3.jar</code>	Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate lets you develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API.
<code>jakarta-oro-2.0.8.jar</code>	The Jakarta-ORO Java classes are a set of text-processing Java classes that provide Perl5 compatible regular expressions, AWK-like regular expressions, glob

Name	Purpose
	expressions, and utility classes for performing substitutions, splits, filtering filenames, etc.
jakarta-regexp-1.4.jar	Jakarta Regexp is a 100% Pure Java Regular Expression package
javachart.jar	Graphing and charting functionality
jaxen-1.1-beta-12.jar	jaxen provides a single point for XPath expression evaluation, regardless of the target object model, whether its dom4j, JDOM, DOM, XOM, JavaBeans
kcServlet.jar	KavaChart's Custom Tag Library will let you quickly and easily generate all sorts of charts based on your data sources. These tags provide a framework that simplifies page design and maintenance, while automatically optimizing the process of generating data driven graphics.
log4j-1.2.13.jar	Log4j is a fast and flexible framework for logging application debugging messages.
proxool-0.8.3.jar	Proxool is a Java connection pool.
quartz-1.5.2.jar	Quartz is a full-featured, open source job scheduling system that can be integrated with, or used along side virtually any J2EE or J2SE application - from the smallest stand-alone application to the largest e-commerce system.
serializer-2.7.0.jar	Xalan library. Containing the Serializer component of Xalan-Java
spring.jar	Spring Framework is the leading full-stack Java/JEE application framework. Spring delivers significant benefits for many projects, increasing development productivity and runtime performance while improving test coverage and application quality.
velocity-1.4.jar	Velocity is a Java-based template engine. It permits anyone to use a simple yet powerful template language to reference objects defined in Java code.
velocity-tools-1.1.jar	VelocityTools is a collection of Velocity subprojects with a common goal of creating tools and infrastructure for building both web and non-web applications using the Velocity template engine.
xalan-2.7.0.jar	Xalan-Java is an XSLT processor for transforming XML documents into HTML, text, or other XML document types.
xercesImpl-2.7.1.jar	Xerces library. All the parser class files that implement one of the standard APIs supported by the parser.

Name	Purpose
<code>xml-apis-2.7.1.jar</code>	Xerces library. All the standard APIs implemented by the parser.
<code>xpp3_1_1_2.jar</code>	Xml Pull Parser (in short XPP) is a streaming pull XML parser and should be used when there is a need to process quickly and efficiently all input elements (for example in SOAP processors).

/meta-inf directory

Name	Purpose
<code>application.xml</code>	XML deployment descriptor for contents of the EAR file.
<code>manifest.mf</code>	Manifest listing the contents of the EAR file.

Components of war-sample.war

enrollment Directory

The **enrollment** directory contains HTML, images, JSPs, and scripts for an example implementation of the Oracle user management framework. For more information and a complete listing of files in this directory, see the *SDK Guide for Oracle Siebel eStatement Manager*.

TIP: The Sample enrollment implementation is a simplified flat model using Common Directory Access (CDA). Most custom implementations want to use **Train**.

META-INF Directory

The **META-INF** directory contains `manifest.MF`, which lists all the components in the EAR file.

user/jsp Directory

The **user/jsp** directory contains JSPs for content access to line item detail, dispute and annotation. For more information and a complete listing of files in this directory, see the *SDK Guide for Oracle Siebel eStatement Manager*.

WEB-INF Directory

The **WEB-INF** directory contains XML deployment descriptors, compiled Java class files, and executable .JAR files that support eStatement Manager.

<root>

Name	Purpose
<code>ibm-web-bnd.xml</code> (WebSphere only)	XML deployment descriptor specific to the IBM WebSphere application server.
<code>sun-j2ee-ri.xml</code>	XML deployment descriptor specific to Sun systems.
<code>web.xml</code>	XML deployment descriptor specific to your Web application's WAR file.
<code>weblogic.xml</code> (WebLogic only)	XML deployment descriptor specific to the BEA WebLogic application server.

/class

This directory contains compiled Java class files for eStatement Manager APIs in the `/com` subdirectory. For more information on files in this directory, please see the *eStatement 3.4 SDK API Specification JavaDoc*.

For *WebLogic* web applications, this directory also contains precompiled JSP class files in the `/jsp_servlet` subdirectory. These pages take advantage of the JSP Compiling feature of WebLogic. Since WebLogic compiles JSPs on the fly, precompiled JSPs load faster when first accessed; if the JSP classes don't exist, WebLogic compiles them at runtime. For more information on compiling JSPs with WebLogic, please see the BEA guide to Using the WebLogic JSP Compiler online at <http://edocs.bea.com/wls/docs92/webapp/reference.html>.

TIP: You can configure WebLogic Server to precompile JSPs by setting the `precompile` parameter to `true` in the `<jsp-descriptor>` element of `weblogic.xml`. See BEA documentation for details.

/lib

This directory contains utility files used by Web application components.

Name	Purpose
<code>edx_servlet.jar</code>	Service JAR file containing Java classes for Web APIs into eStatement Manager. Used by all modules. This file is required in the <code>/lib</code> directory for each WAR file; it is also included at the EAR level for convenience when building a new Web application. Sample Web applications include <code>edx_servlet.jar</code> in both <code>*.ear/lib</code> and <code>*.ear/*.war/web-inf/lib</code> .
<code>javachart.jar</code>	Service JAR file containing KavaChart charting utility shipped with eStatement Manager. Used by Creating Custom Charts . For more information about KavaChart, see http://www.ve.com/ .

Name	Purpose
<code>xtags.jar</code>	Service JAR file containing XTags custom tag libraries (taglibs) for working with XML in JSPs. Used by Content Access and Line Item Dispute and Annotation For more information about XTags and taglibs, see http://jakarta.apache.org/taglibs/doc/xtags-doc/intro.html

7

Appendix B: Components of eStatement Manager Application

Components of National Wireless Dataset

The National Wireless sample application files are located in the **<EDX_HOME>/samples/NatlWireless** directory. For more information about working with National Wireless, please see the *Data Definition (DefTool) Guide for Oracle Siebel eStatement Manager*.

Filename	Purpose
NatlWireless.txt	Sample data input file (source).
DetailExtractor/edx-DE-table.xsd	Custom template for validating XML input to the Detail Extractor job.
DetailExtractor/summary_info.xml	Customizable input file to the Detail Extractor batch job containing business logic rules for extracting summary data as XML.
DetailExtractor/summary_info.xsl	Customizable input file to the Detail Extractor batch job containing XSLT stylesheet rules for presenting extracted XML summary data.
XMLQuery/annot_sql.xml	Customizable input file to the XML Query Web view containing SQL statements for extracting annotation detail from database tables.
XMLQuery/detail_sql.xml	Customizable input file to the XML Query Web view containing SQL statements for extracting line item detail from database tables.
XMLQuery/dispute_sql.xml	Customizable input file to the XML Query Web view containing SQL statements for extracting dispute detail from database tables.
XSLTDownload/summary_info_csv.xsl	Customizable input file to the XSLT Download batch job containing XSLT stylesheet rules for downloading extracted XML summary data as comma-separated values (CSV).
images/B2B_cust_care_agent.gif	Image file
images/B2B_DSLBanner.jpg	Image file
images/B2B_limo_ad_small.gif	Image file
images/B2C_FreeCallingCardSmallPic.gif	Image file
images/B2C_hawaii_promo.gif	Image file
images/Bk2Summary.jpg	Image file
images/edocslogo.gif	Image file
images/LDPhoneCard.gif	Image file
images/Loc_Summary.alf	Contains the customized rules and business logic for mapping and presenting the extracted data in the detail statement view NW_LocSummary.
images/LocalSummary.jpg	Image file
NatlWireless.alf	Contains the customized rules and business logic for mapping and presenting the extracted data in an online summary statement.

Appendix B: Components of eStatement Manager Application Datasets ■ Components of National Wireless Dataset

NatlWireless.ddf	Contains the customized rules and business logic for extracting each field from each page of the original statement file for presenting different views of the data in online statements.
NatlWireless.htm	HTML template for presenting the summary statement view named HtmlDetail.
NatlWireless.tok	Sample file for the Def2TOK utility.
NW_Email.alf	Contains the customized rules and business logic for creating notification email.
NW_Email.htm	Example layout for notification email (its use is determined by business logic).
NW_EmailAlternate.htm	Alternate layout for notification email (its use is determined by business logic).
NW_LocSummary.alf	Contains the customized rules and business logic for presenting the detail statement view NW_LocSummary.
NW_LocSummary.ddf	Contains the customized rules and business logic for extracting each field from each page of the original statement file to present the detail statement view NW_LocSummary.
NW_LocSummary.htm	HTML template for presenting the detail statement view NW_LocSummary.
images/NW_Logo_B2B.jpg	Image file
images/NW_Logo_B2C.jpg	Image file
images/NW_Logo_commercial.jpg	Image file
images/NW_Logo_consumer.jpg	Image file
images/PaymentBannerAd.gif	Image file
LocalLineSummary_0.properties	Charting properties file.

Index

A

Annotation, 47
application.xml, 29

C

CDA, 11, 12, 41, 47
Content Access, 47
Context Root, 27
Creating a new application,
19
Customer Self-Service, 7

D

Deploying J2EE eStatement
Manager Applications to
WebLogic Server, 15
Dispute, 47

E

ear-eStatement.ear, 9
ear-sample.ear, 11, 28, 47
ear-training.ear, 12
ear-umfsample.ear, 12
eaSuite, 7
ejb-enrollment-cda.jar, 27

J

JNDI Names, 27

N

New application
creating, 19

S

Sample, 11, 28
sun-j2ee-ri.xml, 30

T

Training, 11

U

User, 47

V

Validation, 42

W

war-sample.war, 34, 52
web.xml, 35
weblogic.xml, 37
weblogic-ejb-jar.xml, 30, 31