SDK and Installation Guide for Oracle Siebel eaAssist

Version 4.7

Date Published May 31, 2007

ORACLE

Contents

1 Preface

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 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

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

The Installation and Configuration Guides describe how to install eaSuite, configure the third-party platforms that support the eaSuite production environment, and deploy eaSuite J2EE web applications.

This guide provides basic concepts and tools for using and customizing the eaAssist application in conjunction with the eStatement Manager SDK modules. eaAssist provides a GUI interface to allow CSRs to access customer information stored in an eStatement Manager application. It also allows developers to write custom CSR applications against eStatement Manager applications. You will need to use the APIs described in this guide and the eStatement Manager SDK to develop and deploy such custom CSR applications. The eaAssist SDK and Installation manual as well as the eStatement Manager SDK are intended for senior web application developers. It assumes in-depth understanding of and practical experience with:

- eStatement Manager 4.7 system architecture, installation, deployment, application design, and administration
- Java 2 Enterprise Edition (J2EE), including Enterprise JavaBeans (EJB), servlets and Java Server Pages (JSPs)
- Packaging and deploying J2EE applications
- Directory services including the Java Naming Directory Interface (JNDI) and the Lightweight Directory Access Protocol (LDAP)
- Internet technologies, including HTML and XML, web server administration, and web browsers

Related Documentation

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

Print Document	Description
Installation Guide for Oracle Siebel eStatement Manager	How to install eStatement and configure it in a distributed environment.
Data Definition (DefTool) Guide for Oracle Siebel eStatement Manager	How to create Data Definition Files (DDFs) for use in indexing your application and extracting data for live presentment.
Presentation Design (Composer Guide) for Oracle Siebel eStatement Manager	How to create Application Logic Files (ALFs) to present statement data for dynamic online display.

Print Document	Description
Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager	How to deploy and customize the J2EE applications provided by eStatement Manager. This guide also describes how to deploy the Sample application provided by eStatement and how to validate that it is set up correctly by running a job through your installed eStatement environment.
Oracle Siebel eStatement Manager 4.7 Release Notes	

2 Installing and Configuring eaAssist

This chapter provides the steps required to install eaAssist. Some of the topics this chapter will cover are:

- System Requirements
- Supported Stacks
- Installing eaAssist on the Application Server using InstallAnywhere
- Deploying eaAssist EAR files
- Configuring the eStatement Manager CDA Database
- Verifying the Install

System Requirements

eStatement Manager 4.7 and its required software components must be installed before you can install eaAssist. See the Oracle Siebel eStatement Manager Installation and Configuration Guide for more information about installing eStatement.

Supported Stacks

The following stacks are supported for eaAssist:

Operating System	Application Server	Database
Solaris 10	BEA WebLogic 9.2	Oracle10g R2
Windows 2003	BEA WebLogic 9.2	Oracle10g R2
Linux	Oracle Appserver R3	Oracle10g R2
Solaris 10	IBM WAS 6.1	Oracle10g R2

Installation Steps

The steps required to implement eaAssist are:

1. Install eStatement Manager.
2. Install eaAssist on the Application Server as an add-on component to eStatement Manager.
3. Move the eaAssist configuration file to the eStatement Manager location.
4. Configure the eStatement Manager CDA to include enrollment information for the CSR based on your application.

See the Oracle Siebel eStatement Manager Installation and Configuration Guide for more information about installing eStatement. See the eStatement SDK Guide Customizing eaAssist for information about customizing the default sample CSR pages and CDA database to fit your application needs.

Installing eaAssist on the Application Server using InstallAnywhere

The installation procedures in this guide show eaAssist being installed using the InstallAnywhere GUI. You can choose one of two installation modes to install eaAssist with InstallAnywhere:

- GUI Mode (default)
- Console Mode

The first installation procedure in this section shows eaAssist being installed using the InstallAnywhere GUI Mode, which requires you to provide information on several screens about the eaAssist components you want to install and their location. By default, eaAssist is installed in C:\eaAssist. You can change the default installation directory when prompted during the installation procedure.

Console Mode is an interactive character-based installation in which you are prompted to respond to several installation questions during the installation in the command line.

To install eaAssist in GUI Mode

1. Run the executable to invoke the InstallAnywhere GUI:

   ```
   Assistin.exe
   ```

2. Continue with the wizard as you do when installing eStatement Manager from the IA kit.

To install eaAssist in Console Mode

1. Navigate to the InstallAnywhere directory for your platform and run the command to invoke InstallAnywhere, using the –i console flag. For example:

   ```
   Assistin.exe –I console
   ```

   InstallAnywhere displays the banner:

   ```
   Preparing CONSOLE Mode Installation
   ```

2. Respond to each prompt to proceed to the next step in the installation. If you want to change information entered in a previous step, type back.

   A successful installation displays the message:

   ```
   Congratulations! <Application Name and Version> has been successfully installed
   to:

   C:\eaAssist
   ```

   where c:\eaAssist is the default home directory.

After you complete the steps listed above steps, the eaAssist directory structure should contain the following files relevant to configuring eaAssist:

```
eaAssist\
          config\
                edx_eaassist.config.bat (edx_eaassist.config for UNIX)


          db\
                create_schema
                config_tool.bat
                config_tool


          J2EEApps\
                     oracleAS\
                         ear-service.ear
                     weblogic\
                         ear-service.ear
                     websphere\
                         ear-service.ear
                         Deployed_ear-service.ear


          samples\
                eaSample\

                       J2EEApps\
                              oracleAS\
                                     ear-sample.ear
                              weblogic\
                                     ear-sample.ear
                              websphere\
                                     ear-sample.ear
                                     Deployed_ear-sample.ear

       jre\ <jre related files>


       Uninstall\ < eaAssist Uninstall related files>

       pkgUtil\
                build.xml
                package.properties
```

To configure eaAssist in FULL Mode

1 Extract the `ear-service.ear` file located in `ASSIST_HOME\J2EEApps\<app-server>\ear-service.ear` where `ASSIST_HOME` is the eaAssist home and `<app-server>` is the application server.

NOTE: By default, the `ear-service.ear` file is in LITE mode.

2 Extract the `war-service.war` file.

3 Edit file `WEB-INF/classes/com/edocs/service/display/csrLinks.properties` and change the property 'mode' with value 'FULL'

Ex: - mode = FULL

4 Repackage `war-service.war`.

5 Repackage the `ear-service.ear`.

NOTE: You can use JDK JAR utility or any other utility for extracting and repackaging.

Deploying the eaAssist EAR Files

The two ear files provided by eaAssist (ear-sample.ear and ear-service.ear) must be deployed to your application server.

NOTE: You cannot deploy the eStatement Sample application and the eaAssist Sample application in the same domain. This is because eaAssist Sample is generated using the eStatement Sample application. Both applications contain the same ejbs with same JNDI name. The context root of the web component is also the same. By changing the JNDI names/context root and its references, it is possible to customize the ea Assist Sample to work in the same application server domain.

eaAssist application ears can be found under:

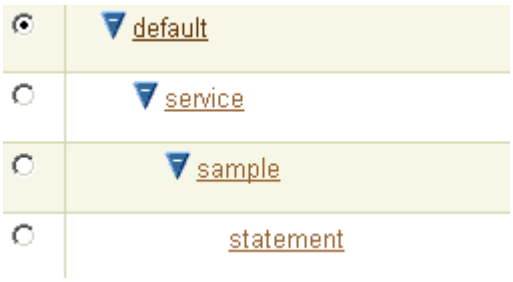
```
J2EEApps\
                oracleAS\
                    ear-service.ear
                weblogic\
                    ear-service.ear
                websphere\
                    ear-service.ear
                    Deployed_ear-service.ear
```

Sample application ears can be found under

```
samples\
          eaSample\
                J2EEApps\
                       oracleAS\
                            ear-sample.ear
                       weblogic\
                            ear-sample.ear
                       websphere\
                            ear-sample.ear
                            Deployed_ear-sample.ear
```

Select the ears based on your application server.

CAUTION: When deploying applications on Oracle application server, make sure you use the following hierarchy:



In this example, 'service' is the root (most parent) application.

Configuring the eStatement CDA Database

The `create_schema` file contains the CDA Client Tool commands to modify the default CDA database provided by eStatement to run eaAssist. To run the `create_schema` script,

Go to the location where the `create_schema` script resides (by default eaAssist\db).

To configure the eStatement Manager CDA Database on Windows

1. Edit the "`config_tool.bat`" file to include eStatement, app server and database information.
2. Open an MS-DOS Command Prompt window and run the "`config_tool.bat`" script.

To configure the eStatement Manager CDA Database on UNIX

1. Run the "`config_tool`" shell script.
2. Provide the eStatement, app server and database information.

Verifying the Installation

After installing and configuring eaAssist and confirming WebLogic (your app server) is running, you should be able to access eaAssist from your browser with the following HTTP address:

```
http//<host>:<port>/eaAssist
```

Where <host> corresponds to the system running application server and <port> corresponds to the port that is running the application.

For example:

```
http//leopard:7001/eaAssist
```

The eaAssist Login page appears.

The initial administration User ID and Password provided for eaAssist is admin and oracle. You should use it to log in initially and later change the password.

Repackaging Third-Party Libraries

After installing and configuring eaAssist you can package the third-party libraries. The following are required to package the third-party libraries:

- eStatement Manager/ePayment Manager/eaAssist must be installed.
- JDK 1.5 should be installed.
- ANT 1.6.5 or later version should be installed.

CAUTION: Make sure Java and ANT paths are properly set.

Open Source Requirements

The following required open source library binaries are not distributed with the product. For more information regarding how to obtain and install these components, refer to add-section-reference-here.

- Ant 1.6.5 is required to execute the supplied Ant database scripts.
- Hibernate 3.1.3 is required for high performance object/relational persistence and query services.
- C3p0 0.90 is required for JDBC3 connection.

Configuring the Environment for ANT and JDK

Operating System	ANT and JDK Configuration
Unix (Solaris/Linux)	Ex:- ANT_HOME=/opt/apache-ant-1.6.5 JAVA_HOME=/opt/jdk1.5.0_04 export ANT_HOME export JAVA_HOME PATH=$ANT_HOME/bin:$JAVA_HOME/bin:$PATH export PATH
Windows	Ex:- Set ANT_HOME=C:/apache-ant-1.6.5 Set JAVA_HOME= C:/jdk1.5.0_04 Set PATH=%ANT_HOME%/bin;%JAVA_HOME%/bin;%PATH%

Downloading and Installing Third-Party Libraries

Download and install the following third-party libraries:

CAUTION: Exact version is required.

Third-Party Library	URL
Hibernate 3.1.3	http://www.hibernate.org/
C3PO 0.9.0	http://sourceforge.net/projects/c3p0/

The library names and installation instructions for each platform are as follows:

JAR File Name (lined to download site)	OS	File Name after Downloading	Tool/Commands to Install
hibernate-3.1.3.jar	UNIX	hibernate-3.1.3.tar.gz	1. gzip -d hibernate-3.1.3.tar.gz 2. tar –xvf hibernate-3.1.3.tar
hibernate-3.1.3.jar	Windows	hibernate-3.1.3.zip	Use Winzip
c3p0-0.9.0.jar	UNIX	c3p0-0.9.0.bin.tgz	1. gzip -d c3p0-0.9.0.bin.tgz 2. tar -xvf c3p0-0.9.0.bin.tar
c3p0-0.9.0.jar	Windows	c3p0-0.9.0.bin.zip	Use Winzip

NOTE: These libraries have to be installed in a server, which can be accessed by eaSuite 4.7 application server components.

Configuring Property Files

While defining paths for windows platform in the 'package.properties' files, use one of the following standards.

- Using forward slash as; `EDX_HOME=C:/eStatement`
- Using 2 backslashes for each single backslash in the path as;

  `EDX_HOME=C:\\eStatement`

In Solaris platform, Solaris standard need to be used while defining paths.

Example: - `EDX_HOME=/opt/eStatement`

Packaging eaAssist

To Change the Property Files for eaAssist

1 Edit the `ASSIST_HOME/pkgUtil/package.properties` property file as follows:

Property File Name	Location	Property Name	Value
package.properties	ASSIST_HOME /pkgUtil	ASSIST_HOME	/opt/eaAssist
		HIBERNATE_JARFILE_LOC	/opt/hibernate-3.1
		C3P_JARFILE_LOC	/opt/c3p0-0.9.0/lib

`ASSIST_HOME` is the eaAssist Home.

2 Edit the file 'package.properties' under 'pkgUtil' folder in eaAssist home.

3 Set the ASSIST_HOME to eaAssist home.

4 Set the property value HIBERNATE_JARFILE_LOC with the Hibernate installation location. If you have downloaded hibernate jar file some other way, then provide up to the folder location where 'hibernate3.jar' resides.

```
Ex: -

/opt/hibernate-3.1/hibernate3.jar → HIBERNATE_JARFILE_LOC=/opt/hibernate-3.1
```

5 Set the property value C3P_JARFILE_LOC with the location where the c3p0-0.9.0.jar file resides.

```
Ex: -
/opt/c3p0-0.9.0/lib/c3p0-0.9.0.jar → C3P_JARFILE_LOC =/opt/c3p0-0.9.0/lib
```

To Package the Third-Party Libraries for eaAssist

1 Navigate to ASSIST_HOME/pkgUtil .

2 Invoke ANT(without arguments).

```
Ex: -
cd /opt/eaAssist/pkgUtil

ant
```

ANT script will complete the repackaging task and display successful message

Failure Recovery

There can be several reasons for the 'BUILD FAILED' error message during the ant execution. The reasons include the following:

- Incorrect package properties file:
  - PRODUCT_HOME is incorrect

    e.g. for ePayment there should be a valid path entry like, PAYMENT_HOME=/opt/ePayment

    If the path is invalid or setting wrong PRODUCT_HOME like EDX_HOME=/opt/ePayment will cause an error.
  - Either HIBERNATE_JARFILE_LOC or C3P_JARFILE_LOC is incorrect.
  - Either the property HIBERNATE_JARFILE_LOC or the property C3P_JARFILE_LOC is not defined at all.
- Unavailability of JAR files in the specified locations.
- JAR file names are incorrect. Expected jar files are:

  hibernate3.jar and c3p0-0.9.0.jar

N.B. Users may get 'BUILD SUCCESSFUL' message without really happening any repackaging if they remove the PRODUCT_HOME property from the package.properties file.

- System crash during the ant execution.
- Lack of free disk space.

To Recover from Build Failure

1. Troubleshoot the possible causes identified in the Failure Recovery section.
2. Re-invoke ant target (no manual removal of partially built components or temporary directories is required).

3 Using eaAssist

This chapter describes the default pages provided by eaAssist to manage CSR tasks. Some of the topics this chapter will cover are:

- Logging into eaAssist
- Searching for customer information
- Using the main Customer Service page
- Defining CSR account information

It is expected that these pages will be customized for a particular client depending on the application requirements. The next chapter describes such customizations, but this chapter does indicate areas where the default behavior will most likely change.

Logging Into eaAssist

To begin using eaAssist, you must bring up the Login page by specifying the URL for eaAssist on your browser. When you installed eaAssist using the instructions in its Installation and Configuration Guide, you would have defined this URL value. It has the following syntax:

```
http://<server-name>:<port-number>/eaAssist
```

For example:

```
http://duster:7001/eaAssist
```

A CSR user that has been added to a defined group can log in. If the CSR belongs to a group that has access to a single application (DDN), the CSR is routed to the Customer Search page for that application (described later in this chapter). Otherwise, the CSR is shown the Application Selection page in order to choose the correct one.

For example:

eaAssist is initially configured with an Administration account to define CSR users and groups. By default, the User ID is admin and Password is Oracle for this account. It belongs to the Admin group and has full privileges, so all applications are accessible and it always goes to the Application Selection page after login.

After a CSR selects a DDN, they are routed to the Customer Service page.

> **TIP:** If at any time your session times out, you will be returned to the Login page to re-enter your User information. You will always return to the current page you requested.

About the Banner Frame

A banner frame is visible on all eaAssist pages after login. It has the eaAssist logo and will have various buttons available depending on the page being displayed and the privileges of the eaAssist user.

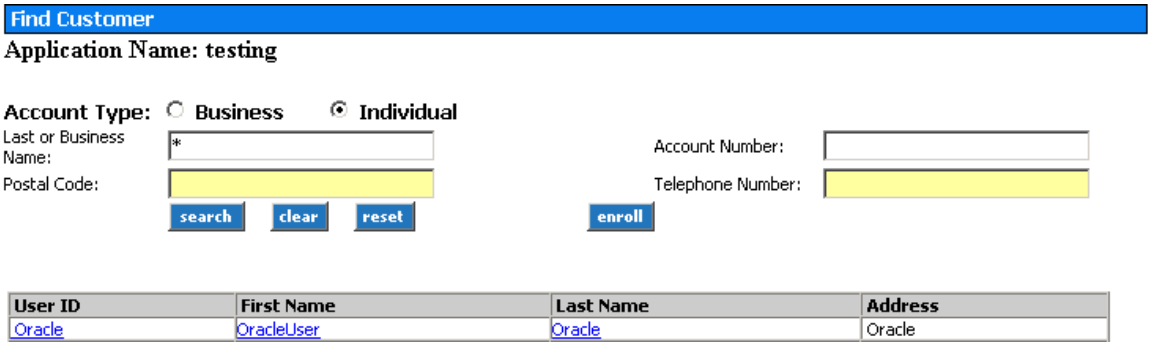
The banner will always have the Help button and to the rightmost end the Logout button. The Help button pops up a new window with a placeholder file where developers can customize their own HTML help.

If the CSR who has logged in also has Admin privileges, the Admin button will also appear on the banner.

Searching for Customer Information

The Customer Search page allows the CSR to enter one or more values in the various text fields provided to retrieve enrolled customers.

For example:



The important concept to understand at this point is that eaAssist works in conjunction with an eStatement Manager application, and can only access information about customers that have enrolled in that application. eaAssist provides the Enroll link for the specific purpose of allowing the CSR to directly enroll the customer if they have not already done so. By default it points to the generic eStatement Manager enrollment page, but it is customizable through the LinkPages properties file described in the next chapter. The example implementation is based on a CDA-based implementation of IAccount.

The default search fields are:

- Account Type where you must choose Individual or Business (CDA schema name = acctType)

- Last or Business Name (CDA schema name = sn)
- Account Number (CDA schema name = accountNumber)
- Postal Code that captures state and city information so it is more concise (CDA schema name = postalCode)
- Telephone Number (CDA schema name = telephoneNumber)

The fields available for search purposes correspond to the eStatement Manager index fields defined for an application and are expected to be customized (also part of the LinkPages property file). It is recommended that developers create a custom version of this page using the JSP and related files, and that they save the customized version under the custom sub-directory in the EAR.

- eaAssist accepts the following syntax as search criteria:
- Wild card character "*" may be used to specify a match.
- Null or blank fields act as a "*", matching everything.
- The search is case sensitive.

For example, a search pattern for "Doe" in the Last name field can be one of the following: "D*", "Do*", or "Doe".

After specifying the value, click on Search to begin, or Clear to reset the page (the Search button on the banner frame also resets the page).

Customer Search Results

All customers that meet the search criteria are retrieved from the database. The search results are presented below the search criteria. For "individual" type accounts, the results table presents:

- User ID
- First and last names
- Address

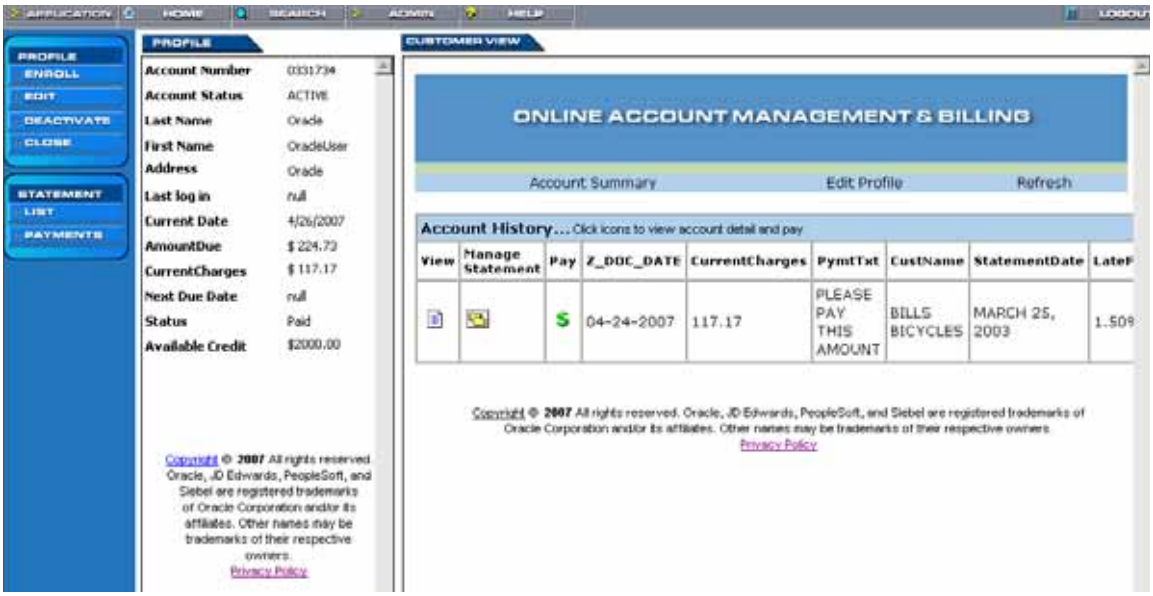
For "business" type accounts, instead of the first and last names, the business name is shown. The CSR can then select the customer from the list to help.

Using the Main Customer Service Page

After selecting a customer, eaAssist presents the main customer service page where the CSR can view and change customer statement information. There are four main frames that are presented:

- Banner
- Left navigation
- Customer profile
- Customer view

For example:



The banner contains the same buttons described earlier, plus the Home button, which returns you to the initial view of the page. Its purpose is to bring the CSR back to the starting point after going into several additional views.

The left navigation pane provides the following buttons to handle customer profile information:

- `Enroll`: provides the same functionality as the Enroll link as described earlier in this chapter, except the enrollment page appears in the customer view pane.
- `Edit`: allows the CSR to edit the customer profile of the current customer.
- `Deactivate`: marks the current enrolled customer as inactive, but it does not remove the entry from the enrollment database tables.
- `Close`: returns the CSR to the Customer Search page.

It also provides the following buttons for viewing statement information:

- `List`: displays the list of statements processed by eStatement Manager and available to the customer and CSR.
- `Payment`: if your application also has ePayment enabled, you can link to that information through this button.

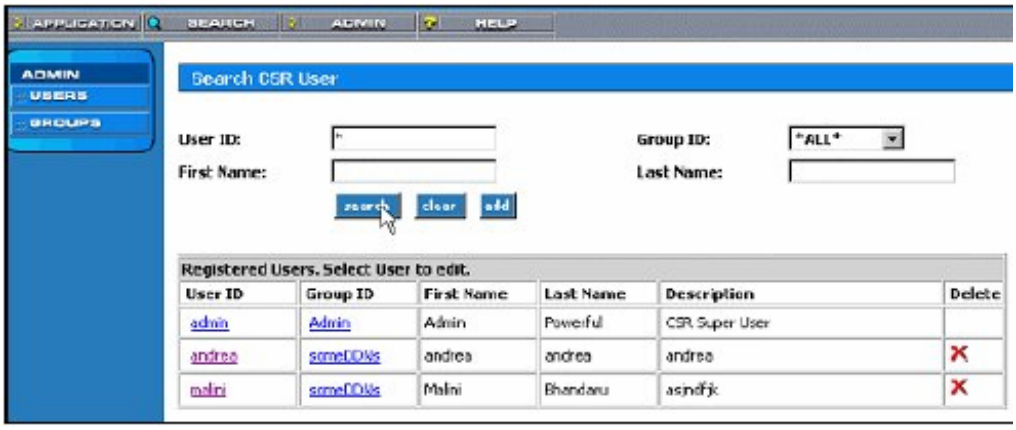
The customer profile pane is another customizable page that displays customer statistical information. The information comes from customer enrollment, activity logs, and previously indexed fields in the eStatement Manager application. All of this information is stored in the eStatement Manager database, so the statement information must have been already indexed.

The customer view pane contains the current detail information about a customer statement. Initially, this pane should carry a CSR tailored view of the statement so as to present just the relevant information a CSR is authorized to see (for example, a CSR may not be allowed to view credit card information). It is expected that a developer will customize this view.

However, the initial default for eaAssist as shown above is the History List information provided by eStatement Manager for a customer. In most cases the CSR will be shown a specific customer statement (such as the current one) and then use the List button to view the History List information.

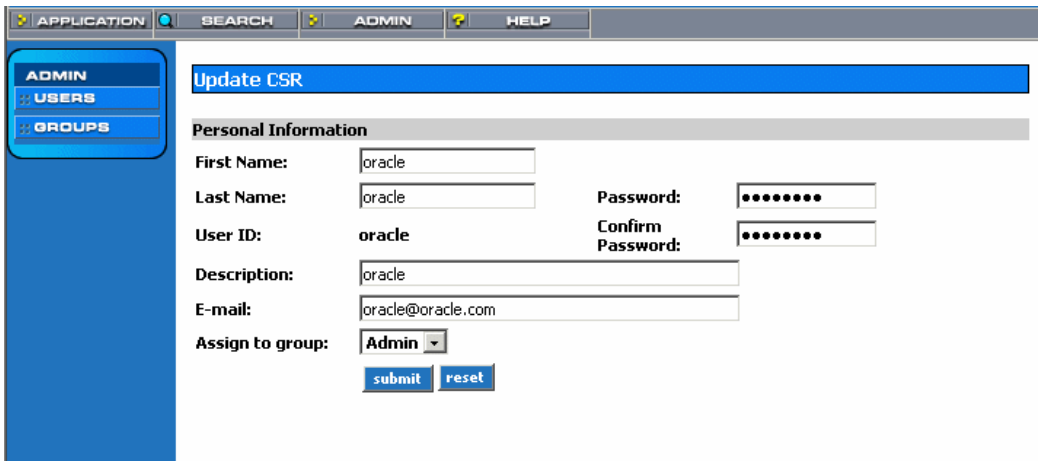
Defining CSR Account Information

CSRs belonging to the Admin Group or other group with the Admin Access Control privileges can search the CSR database, view a CSR profile, edit a CSR profile, and add/delete CSR personnel. After the CSR logs in, this Admin privilege is evident by the Admin button that appears in the banner. Clicking on that button brings up the Search CSR User page. For example:



As with previous search pages, you can enter values for any of the search fields and click the Search button to find any matches. If you leave the fields blank, eaAssist will find all entries. In the above example, eaAssist has found two registered users, the default admin user and a specific CSR user that has been added (only the admin user is initially stored with eaAssist upon installation).

You can delete a CSR user from eaAssist by clicking on the delete icon at the end of the row. However, eaAssist will not allow you to delete a registered user if you have logged in under that User ID. If you click on one of the user names, the Update CSR page appears for that selection:



If you need to add a CSR, you can click on the Add button from the Search CSR page to get to the Add New CSR page with blank fields to enter information:

On either page, you can modify or add the CSR information and submit it. Special characters are not accepted for the input fields such as the line feed or carriage return. Also, the email address should be verified for correct format.

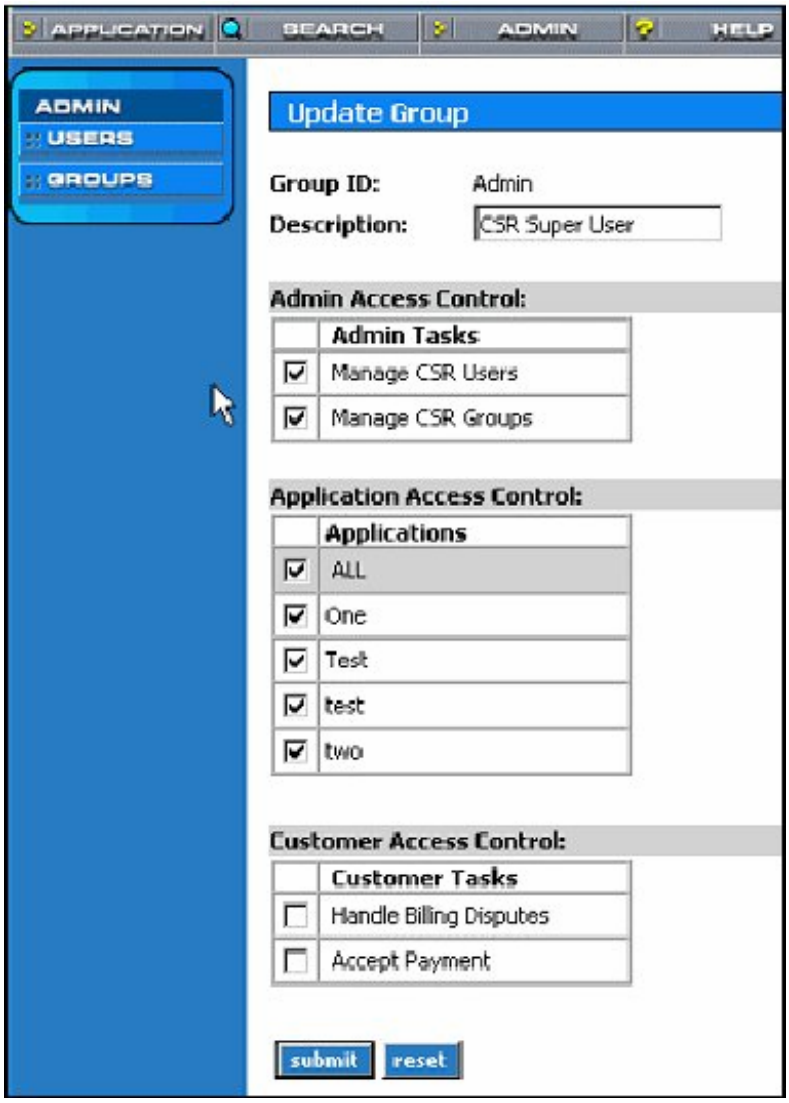
> **TIP:** eaAssist only renders a blank password, not even duplicating the encrypted password to the Admin CSR viewer of this page (of course a custom implementation of eaAssist can change this).

Creating and Assigning CSR Groups

The last field on the above pages allows you to assign the CSR to a particular group, so that specific access controls can be assigned to a wide variety of CSRs. On the left navigation pane are two Admin buttons that allow you to switch between defining a CSR or a Group. Clicking on the Users button brings you back to the Search CSR page. Clicking on the Groups button takes you to the Search CSR Groups page. This page is similar to the previous Search CSR page, but instead you can search for CSR groups to modify.

You can enter a Group ID or Description of the Group you wish to search for. For example, enter Admin in the Group ID field to find the CSR Super User as noted in the Description. If you leave the fields empty and click the Search button, all groups are returned.

You can delete a CSR group from eaAssist by clicking on the delete icon at the end of the row. However, eaAssist will not allow you to delete a group if you have logged in with a User ID assigned to that group. Click on a Group ID in the results table to go to the Update Group page. For example:



Or click on the Add button to go to the Add New Group page, which is similar to the Update Group page except that it allows you to enter the Group ID name.

It is on these pages you can create a new CSR Group and assign specific Admin privileges and access to specific applications for CSRs affiliated with a Group. For example, if you want to limit a CSR to access only one application so that they directly go to the Customer Search page for that application upon login, you can just check that application from the list under Application Access Control.

The Admin Access Control options allow you to define the specific Admin Task privileges the group is allowed to make. The Customer Access Control section lists the Customer Tasks the CSR assigned to this group is allowed to do on behalf of a customer.

CSR Account Attributes

The following table describes the attributes of a CSR Account.

Description	CDA Attribute	Type/Length
First Name	*givenName*	Text/32
Last Name	*sn*	Text/64
Password	*userPassword*	Text/15
Description	*description*	Text/255
Email	*mail*	Text/127
Group	*group*	Pre defined

CSR Group Attributes

The following table describes the attributes of a CSR Group.

Description	CDA Attribute	Type/Length
Group Id	*gid*	Text/32
Description	*description*	Text/64
Privilege	*priv*	Multiple values, text, possibly none
Data Definition Name	*ddn*	Multiple values, text, possibly none

4 Customizing eaAssist

This chapter describes how to build a custom EJB application that is based on the generic eaAssist application that you can deploy to the WebLogic 9.2 J2EE platform. This includes creating and modifying JSP, HTML, and Java Class files depending on the complexity of your custom application. The sections in this chapter describe how to change the WAR including its deployment descriptor.

About the eaAssist EAR

Part of getting a custom eaAssist application to work is deploying it as an EAR file (**E**nterprise **A**pplication a**R**chive). When you install and configure eaAssist along with eStatement Manager, it deploys the following EAR files to the *<weblogic-home>/user_projects/domains/mydomain/applications* directory, where *<weblogic_home>* is the directory location where you installed WebLogic:

Application Files	Description
ear-eStatement.ear	Command Center application
ear-sample.ear	Sample user application
ear-training.ear	Custom hierarchical application
ear-payment.ear *ear-eapost.ear*	Skeleton auxiliary applications
ear-service.ear	Generic eaAssist application

This is the location where you will deploy your custom application.

Use the eaAssist EAR file as the basis for your new custom CSR application as it contains all the necessary components to get started. You should copy that file to a working directory so you can begin to modify its contents and add any custom files. For example:

1. Create a working directory somewhere on your file system.
2. Copy the eaAssist EAR file (*ear-service.ear*) from

   <eaAssist_home>/J2EEApps/weblogic to your working directory, where *<eaAssist_home>* is the eaAssist installed location.
3. Extract the files out of the EAR file by using the following command from your working directory:

   `jar xvf ear-service.ear`

After the extraction, your working directory should contain the eaAssist WAR file, *war-service.war*.

Delete the *ear-service.ear* file from your working directory as it is no longer necessary.

The WAR file supplied with eaAssist contains the following types of components that can help you build your web application:

- Servlets
- Classes
- HTML pages
- JSPs
- Java scripts
- Images

You can extract the contents of the WAR file using the following command:

```
jar xvf war-service.war
```

However, you should create a temporary subdirectory first to copy the WAR file and move the WAR file to that location before extracting its contents. After the extraction, you can delete the war-service.war file as it is no longer necessary. The relevant files extracted include:

- Class files that are used by the application servlets. You may need to modify these for your specific needs. They reside in the *WEB-INF* directory:
- The deployment descriptors for the WAR file that defines the servlets and EJB references used by the web application. They also reside in the *WEB-INF* directory
  - *web.xml*
  - *weblogic.xml*
  - *sun-web.xml*
- The eaAssist JSPs, Java scripts, and HTML pages located in the admin directory.
- The *index.html*.

The *index.html* file routes a user to the first page of the eaAssist application when it is deployed.

Defining Servlets for Your Web Application

In a web application, the servlets the application uses go within the application structure itself defined for the WAR file. If you create new class files for existing or new servlets, you should create a subdirectory in the *WEB-INF/classes/com* directory to contain them.

If you want to create a new servlet to access a class file, you should:

1. Write and compile your servlet class file as usual.
2. Create a new directory in *WEB-INF/classes/com* such as newapp to contain the class file and place it there.

After you have the class files in place, you need to tell the web application where to find those resources by modifying the deployment descriptor files, *web.xml* and *weblogic.xml*. The *web.xml* file defines the servlet and EJB references used by the web application, as follows:

- For the web application it defines:

- Display name
- Description
- Welcome file

- For each servlet it defines:
  - Servlet name
  - Servlet description
  - Servlet class
  - Servlet parameters
- For servlet mappings it defines:
  - Servlet name
  - URL pattern
- For each EJB used it defines:
  - Reference name
  - Reference type
  - Home and remote interfaces

The *weblogic.xml* file defines the JNDI mappings for the EJBs used by the web application (each reference must map to one specified in the *web.xml* file).

So, if the objective is to create a new custom servlet definition to point to your new class file, you would have to add an entry in the *web.xml* file for that servlet and point it to the new class file you placed in *WEB-INF/classes/com/newapp*.

Defining the Context Name

The first (and required) step to creating your custom application is deciding on a context name. This name will be used as the web-context by which all URL references will be based on, and it will be used as a prefix to the JNDI name references in the descriptor files for your WAR file. It will even be used in the name of the WAR and EAR files. For these reasons, it is easier to be consistent across all your files with this context name.

For the eaAssist EAR file described in the previous chapter, the context name is *eaAssist*. You should search for it when replacing the context name; the next set of steps will describe how to do this throughout the EJB application. This example will use *newcsrapp* as its context name, but you should use your own descriptive name.

Follow these steps to be thorough when replacing the name:

1. Edit the *META-INF/application.xml* file. Replace all occurrences of *eaAssist* with *newcsrapp*.

3. It is expected that your custom application files will replace most of the files currently used by eaAssist in the WAR directories. However, if you reuse one of these files for your application, remember to substitute instances of *eaAssist* with *newcsrapp* in the file (usually when making a URL reference).

4. Edit *WEB-INF/weblogic.xml* to change all instances of *eaAssist* to *newcsrapp*. Do the same for the *web.xml* file.

5. If the context root of the CSR application needs to be changed to "newcsrapp", modify the WEB-INF/sun-web.xml file to reflect the following.

   <context-root>eaAssist</context-root>

   to

   <context-root>newcsrapp</context-root>

Packaging and Deploying the Custom CSR Application

After you make any changes to the contents of the web application, you will need to package them back into a WAR file. For example:

```
jar cv0fm war-newcsrapp.war META-INF/MANIFEST.MF
```

The new WAR file is ready to be moved from the temporary subdirectory up to the parent directory. You should delete the temporary subdirectory from your working directory. The WAR file is now ready for inclusion in the packaged EAR file, as follows:

i. Modify the EAR deployment descriptor file, *META-INF/application.xml*, to include information about WAR changes. It specifies:

- Display name
- WAR file or WEB URL
- Context root (if you did not do this previously)
- Any EJBs (if you added them)

2. Package the application into an EAR file. For example:

```
jar cv0fm ear-newcsrapp.ear META-INF/MANIFEST.MF
```

3. Deploy the EAR file using WebLogic's Console utility. (See the WebLogic documentation for information about this.).

However, in the development phase of the lifecycle of your project, it is tedious to package and re-deploy the application each time you want to test changes you have made to the application. WebLogic allows you to deploy your application in directories mode for your WAR file components so it can pick up your changes instantly without the need to redeploy, stop, and restart WebLogic. For more information about this, consult the eStatement Manager guide *Deploying and Customizing J2EE Applications*.

About the eaAssist CDA Schema

eaAssist extends the default eStatement Manager user management framework that uses the **Common Directory Access** (CDA) interface to emulate the core features of an LDAP service provider. CDA implements the JNDI public interface `DirContext` to map a hierarchical **namespace**

onto a directory. `DirContext` contains methods for examining and updating attributes associated with objects, and for searching the directory.

For detailed information about the eStatement Manager user management framework and how to customize it, you should consult the guide: *User Management Frameworks*.

The CDA Directory Information Tree

The eStatement Manager default CDA schema extends the well-defined LDAP **directory information tree** (DIT), in which each object is a set of attributes. The DIT developer determines which attribute names the object, whether objects may include themselves, and which attributes are required. Basic objects and their rules include the following:

- Countries (`c`) may contain Organizations.
- Organizations (o) may contain Organization Units.
- Organization Units (ou) may nest three levels deep, and may not contain an Organization.

The CDA model supports attributes of the value types `String` and `DirContext`. CDA attribute and value names are encoded to the ISO-8859-1 data standard only. Attribute names are limited to 255 characters and values are limited to 1024 characters, encoded according to the schema.

The eaAssist schema requires additional attribute names that it must add, or **bind**, to the schema before giving a value to an attribute name. These are described later in this section. The actual CDA commands used are `SchemaBind` and `CreateSubcontext`.

Add New Attribute Names with SchemaBind

Before assigning a value to an attribute name in a directory, you must add, or bind, the attribute name to the schema. In the CDA Client, add a new attribute name to the schema with the command **`SchemaBind (sb)`**.

```
sb <attribute-name> <syntax>
```

`SchemaBind` takes two parameters, attribute name and syntax. There are only two valid values for the syntax parameter: **`String`** (default) and **`Distinguished Name`** (DN). You must specify syntax of **`DN`** (upper case required) when adding an attribute of the Distinguished Name type.

To add an attribute named "employee" with the default syntax of **`string`**, issue the following command:

```
Sb employee
```

To add an attribute named "employee" with a syntax of distinguished name, issue the following command:

```
Sb employee syntax DN
```

CreateSubcontext

```
mk name [attribute-name attribute-value ...]
```

Creates a new subcontext name and associates all specified attribute names and values with the new subcontext.

Creating the CSR Enrollment Hierarchy With CDA

Common Directory Access (CDA) supports hierarchical enrollment schemas that nest users in subaccounts. The **`IAccount`** API emulates the JNDI interface **`DirContext`** to afford the flexibility and power of JNDI and LDAP. The **`IAccount`** API replaces JNDI methods that return instances of **`DirContext`** with similar methods that either return context names or reset context state inside the account object.

To create the eaAssist CSR enrollment hierarchies, the *create_schema* file which is located under the *<eaAssist_home>/db* folder is run during installation. The *create_schema* file is used as input to a java process that alters the eStatement Manager CDA database for enrollment information. For information about how to run that CDA script, see the Configuring the eStatement CDA Database section of this document for your platform.

CSR enrollment and privilege information is stored on a separate branch of the CDA database of eStatement Manager. The following are the contents of the *create_schema* file:

```
sb group syntax STRING description "CSR user's group which determines his/her
privileges"

sb gid syntax STRING description "Group ID"

sb status syntax STRING description "Indicates whether a customer's account status
is active or in-active"

sb priv syntax STRING description "a CSR privilege such as manageCSRUsers,
manageCSRGroups, handleDispute, acceptPayment"

sb acctType syntax STRING description "account type is either individual or
business"

sb address syntax STRING description "full address"


mk cn=CSR,o=edocs.com

mk cn=Groups,cn=CSR,o=edocs.com

mk gid=Admin,cn=Groups,cn=CSR,o=edocs.com gid Admin priv manageCSRUsers priv
manageCSRGroups description "CSR Super User" ddn "*"

mk gid=CSR,cn=Groups,cn=CSR,o=edocs.com gid CSR  description "Generic CSR User"
ddn "*"

mk uid=admin,cn=CSR,o=edocs.com userPassword 971119623119E56B uid admin  givenName
Admin sn Powerful mail admin@oracle.com group
"gid=Admin,cn=Groups,cn=CSR,o=edocs.com" description "CSR Super User"
```

The schemas “group”, “gid” and “priv” are essential to define CSR accounts and associate privileges based on their group affiliation.

The schemas “status” and “acctType” are for the sample billing application provided with eaAssist to demonstrate the product. A typical customization is to introduce a schema such as these to meet the specific needs of your application.

Avoiding Race Conditions

The eStatement Manager authentication framework uses session-beans (ISession and IAccount) that as per the EJB specifications are not re-entrant. If a UI page has multiple frames and they must all load by first authenticating, a race condition can occur. This could happen because the pages will try to load in parallel and they would all then be trying to access the authentication data structures that are non-re-entrant.

A solution is to authenticate the container pane and make its child panes load directly without authentication, or at most only one child pane re-authenticate. This is the approach used in the eaAssist CustomerPage Frame set. Only the rightmost pane that frequently needs re-painting is re-authenticated. When two or more panes need to be refreshed, it is done by loading a frameset, and the frameset page is re-authenticated.

To elaborate, the CustomerPage Frameset consists of a Banner, Left Navigation Bar, Customer Profile and Customer View page. The Banner and Left-Navigation load without authentication since they expose only buttons, no private information. The first time the Customer Profile and Customer View frameset loads, the frameset page or container page is authenticated. Most navigation bar button clicks affect only the Customer View Page and that page always authenticates to ensure that the CSR's session has not timed out and that there is a valid CSR viewer. If the Customer Profile page needs refreshing, eaAssist re-loads the frameset and that requires re-authentication.

This race issue must always be kept in mind when dealing with framesets.

Customizing the Links to Other Applications

`LinkPages` is the class that supports linking to various application specific custom pages such as for enrollment, customer search, customer profile, and customer summary information. The purpose is to provide developers of eaAssist an easier way to handle such customizations without having to edit JSPs (or at the very least minimize JSP editing).

The assumption is that the customer application already exists and the eaAssist CSR application needs to tie in to specific pages such as enroll and summary. Customer Search and Profile pages would typically have to be built specifically for the application. For example, a Utility billing application might want to cite on the Customer Profile page the monthly consumption average and year-to-date use. Or if the application is financial, perhaps the Customer Profile page needs to display fund performance or profits and losses.

If the degree of customization required is very high, the Link pages class may not be able to support it, requiring the developer to edit the JSPs.

Custom Links

Custom links are specified by editing the file *WEB_INF/classes/com/edocs/service/display/csrLinks.properties*. The idea behind this file is that on a per application (DDN) basis one can specify customized links and implementations.

The account implementation used to create user accounts by eaAssist in order to assist customers is specified using the property "AcctImpl". The default is "edx/Sample/ejb/CDAAccount" (it is specified without a DDN tag), which is the JNDI name for the implementation used by the sample billing application provided with eaAssist. As a customization example, suppose a custom enrollment implementation is used for the billing application XYZ. The custom implementation is specified in *csrLinks.properties* using the entry:

```
XYZ.AcctImpl = xyzImpl
```

Note that `xyzImpl` must be the JNDI name for the account implementation to be used.

The context prefix to be used to create and retrieve account related information for the XYZ application must be provided also:

```
XYZ.ctxPrefix = xyzPrefix
```

The Left Navigation bar on the Customer Main Page (described in Chapter 2) can be customized in the *csrLinks.properties* file to link to different pages based on the application context. The buttons Enroll, Deactivate, Summary, etc. are links that are retrieved using the pattern just as that used for AcctImpl. That is, the default enroll page, specified using the property "enroll" is:

```
enroll=/Sample/UserEnrollment?app=SubscribeApp&jsp=/enrollment/jsp/user_get_subscr
ibe.jsp
```

If you want a custom enroll page for application XYZ , you can specify it as follows:

```
XYZ.enroll=/eaXYZ/UserEnrollment?app=SubscribeApp&jsp=/enrollment/jsp/user_get_sub
scribe.jsp
```

The same thing applies for the pay, summary, edit, and profile and search links. The only difference is that typically the profile and search links will be to JSP pages in the eaAssist application, since these are typical CSR activities and more often than not a part of the billing application.

The table below shows the default entries in the *csrLinks.properties* file:

Property	Value
AcctImpl	`edx/Sample/ejb/CDAAccount`
deactivate	`/eaAssist/CSCenter?app=CSRMain&submit=deleteAcct&jsp=/admin/jsp/CustomerMainPage.jsp`
summary	`/Sample/User?app=UserMain&jsp=/user/jsp/HistoryList.jsp`
detail	`/Sample/User?app=UserMain&jsp=/user/jsp/Detail.jsp`
edit	`/Sample/UserEnrollment?app=UpdateApp&jsp=/enrollment/jsp/user_get_subscribe.jsp`
enroll	`/Sample/UserEnrollment?app=SubscribeApp&jsp=/enrollment/jsp/user_get_subscribe.jsp`
pay	`/eaAssist/CSCenter?app=CSRMain&jsp=/admin/html/PayHistory.htm`
profile	`/admin/jsp/CustomerProfile.jsp`
search	`/admin/jsp/CustomerSearch.jsp`

Note how some of the links go into a separate web application, in this case Sample while others stay within the eaAssist application such as "search" and "profile". The pay link could have also been specified as */admin/html/PayHistory.htm* that eaAssist uses as a placeholder. The payment link would

depend on whether payment was installed and if so whether a simple or complex enrollment structure is involved.

Links that custom applications will often want to modify are: the enroll page and the summary page. The default set up takes you to the Sample Enrollment page and HistoryList page. If you want to change the name of the links, not just the link itself, then you must edit the JSPs, doing a textual substitution for the link name. Use the generic link retrieval methods in the class `HelpApp` to retrieve the value of the link. We recommend using the link names provided and introducing new link names if your CSR application needs them. You will also have to provide additional roll-over images for the various navigation bars if you would like these new links to be available for quick navigation.

About the Default eaAssist JSPs

If you look at the contents of the *admin/jsp* directory, the JSPs can be divided into three sets:

- Common
- Custom
- Admin

One can also partition the JSPs into those that provide navigational ability and those that display sensitive information. All navigational JSPs are reached in the application more or less directly. Before the selection is displayed that contains sensitive information, the user is authenticated. For authentication and general servicing, all eaAssist requests are gated by the servlet CSRMain.

The navigational JSPs are: *Banner.jsp*, *LeftNavigation.jsp*, *and LeftNavAdmin.jsp*. A login page must never appear in their stead because they are reached directly.

Common JSPs

The JSPs that fall into the common group are shared by most of the other JSPs. Their names typically start with a lower case letter.

JSP	Function
customer.jsp	Defines DDN and uid variables.
nav.jsp	Imports LinkPages.java to facilitate custom page links.
privs.jsp	Obtains the CSR's privileges and makes them accessible to the JSP pages for controlling access to various functionality.
searchHead.jsp	Java class imports necessary for a CDA-based search.
ErrorMsg.jsp	Used to return a standard error message, whose source contains details about the error message as a hidden field.
UserMsg.jsp	Used to return friendly informative messages to the CSR user, typically necessary while configuring the system and something is found to be missing.

It is expected that none of these JSPs will need customization. If this proves not to be the case, do not change or remove functionality that exists today.

Custom JSPs

JSPs whose name starts with the prefix "Customer" are expected to be customized to meet an eStatement Manager deployment more closely or perhaps have variants for each customer application running against a particular eStatement Manager installation.

It is recommended that such page customizations be performed using these JSPs as a template. Java code may need to be added or dropped, but the structure of the JSP and its inclusions should be maintained to avoid problems. Note that form variables must have the prefix "user__" (or "auth__" or "csr__" as the case may be) to be picked up by the backend code that carries out search, authentication and other tasks.

The Customer related pages are summarized in the table below:

Customer	Function
CustomerMainPage.jsp	Four frame page setup: banner, left navigation, customer profile and customer detail.
CustomerProfile.jsp	Customer profile page, typically is customized based on the enrollment data available and the indexed fields in the application.
CustomerSearch.jsp	Search criteria and jsp inclusions for search and their results.
CustomerSearchForm.inc	Form for specifying search criteria. Note the user__ prefix for form fields that are required to be propagated.
CustomerSearchResults.inc	Search result display for the sample billing application, includes conditional logic based on account type.
CustomerDeActivate.jsp	This page handles de-activation of a user account. The default implementation sets the 'password' of the account to a random string and sets 'status' to inactive. This page may typically be customized to meet different requirements.
LeftNavigation.jsp	This page is customized indirectly via the LinkPages class and csrLinks.properties file. Its goal is to provide the CSR navigation capabilities on the main customer page.
Banner.jsp	Provides navigation ability across eaAssist. You may need to customize it for online Help.
OnePage.jsp	Has in-line frames that display CustomerProfile and Customer View pages. The Customer View page is a placeholder that is used to display summary, bill detail and account status information or any other information. It is the main pane that gets re-drawn based on button clicks.

CustomerSearch.jsp

This section describes in greater detail the CustomerSearch.jsp page shipped with eaAssist. It makes use of some common search keys and illustrates how one may develop and customize such search functionality.

The CustomerSearch.jsp actually includes the following JSPs:

- *searchHead.inc* (belonging to the "common" JSP group)
- *CustomerSearchForm.inc*

- *CustomerSearchResults.inc*

When a search has not yet been initiated, the search results are null and neither of the result JSP pages are included. After a search, the appropriate result JSP page is included based on the account type sought. The above structure is illustrated in the code reproduced below for *CustomerSearch.jsp*:

```
<%@ include file="/admin/jsp/searchHead.inc" %>

<SCRIPT language="javascript1.2"
src="admin/scripts/customerSearchInputCheck.js"></SCRIPT>


<body bgcolor="#2675BA" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
onLoad="checkForMsg();">


<table width="100%" border="0">

  <tr>

    <td> </td>

    <td> </td>

    <td> </td>

  </tr>

</table>

<center>

<% Properties searchEnv = HelpApp.getParamsNoTrim(request, "user__");

   String submit = request.getParameter("submit");

%>

<%@ include file="/admin/jsp/CustomerSearchForm.inc" %>

<%

   if ((submit != null) && submit.equals("search")) { %>

      <%@ include file="CustomerSearchResults.inc" %>

<% } %>

</center>

<%@ include file="/admin/html/genericFoot.htm" %>
```

The following *CustomerSearchResults.jsp* page contents are provided to show how the search criteria have been used. Note how matchAttrs is constructed from non-empty search fields. The next act is to search the enrollment database using the search criteria and determine if any matches are returned. For each match the desired fields to display are next sought from the attributes returned. A link is created which when selected takes the CSR to a Customer view.

NOTE: The fields that will be necessary to authenticate the users towards the CSR serving them must be specified using "auth__" as a prefix.

In the account implementation example provided, the fields *auth__role*, *auth__dn* and *auth__uid* must be provided. Of these *auth__role* is CSR and fixed. The other two depend on the customer and created on the fly. For perhaps a custom enrollment model, a certificate may need to be passed in. On the backend, the fields prefixed with "auth__" are used to populate a hashtable that in turn is passed to the IAccount method "impersonate".

Refer to the SDK javadocs about the IAccount interface for further details. Also note the need to URL Encode the values that are to be folded into the **link** , such as `URLEncoder(uidStr)`.

CustomerSearchResults.inc

```
<table width="750" border="0" bgcolor="#FFFFFF"

       cellpadding="0" cellspacing="0">


<%

    IAccount root = null;

try {

    root = HelpApp.createCustomerAccount(ddn);

    String details =
"CSCenter?app=CSRMain&jsp=/admin/jsp/CustomerMainPage.jsp&submit=setAcct"

                      + "&auth__role=CSR"

                      + "&ddn=" + ddn

                      + "&uid=" ;

    Attributes matchAttrs = new BasicAttributes(true);

    matchAttrs.put(new BasicAttribute("ddn", ddn));

    matchAttrs.put(new BasicAttribute("acctType", acctTypeStr));


    if (HelpApp.nonEmptySearchKey(lastNameStr, "*")) {

        matchAttrs.put(new BasicAttribute("sn", lastNameStr));

    }


    if (HelpApp.nonEmptySearchKey(accountNumberStr, "*")) {

        matchAttrs.put(new BasicAttribute("accountNumber", accountNumberStr));

    }


    if (HelpApp.nonEmptySearchKey(postalCodeStr, "*")) {

        matchAttrs.put(new BasicAttribute("postalCode", postalCodeStr));

    }
```

```
    if (HelpApp.nonEmptySearchKey(telephoneNumberStr, "*")) {
        matchAttrs.put(new BasicAttribute("telephoneNumber", telephoneNumberStr));
    }

    boolean isIndividual = "Individual".equalsIgnoreCase(acctTypeStr);
    String [] attrIds = {JNDIAccountAttributes.JNDI_UID,
                         "sn",
                         "givenName",
                         "postalCode",
                         "address"
                        };

    SearchResult[] s = root.search("", matchAttrs, attrIds);
    if (s.length == 0) { %>
      <tr> <td  bgcolor="#CCCCCC">   </td></tr>
      <tr> <td class="font8ptBold" bgcolor="#CCCCCC">
         No matches found! </td> </tr>
      <tr> <td  bgcolor="#CCCCCC">   </td></tr>
      <tr> <td>   </td></tr>
<%  } else { %>
    <tr>
      <td width="3%"> </td>
      <td colspan="3">
        <table border="1" cellspacing="0" cellpadding="0" width="700"
bgcolor="#FFFFFF">
          <tr bgcolor="#CCCCCC">
            <td class="font8ptBold"> User ID</td>
<% if (isIndividual) { %>
            <td class="font8ptBold"> First Name</td>
            <td class="font8ptBold"> Last Name</td>
<% } else { %>
```

```
            <td class="font8ptBold"> Business Name</td>
<% } %>
              <td class="font8ptBold"> Address</td>
            </tr>

<%
        // using last name to sort
        s = HelpApp.sortSearchResults(s, "sn");
        for (int i = 0; i<s.length; ++i) {

            String uidStr = null;
            String uidStrEncoded = null;
            String firstName= null;
            String lastName = null;
            String address = null;
            String postalCode = null;
           String theLink = null;
           String dnEncoded = URLEncoder.encode(s[i].getName());
          //out.println("<!-- fullname = " + nameInNamespace +  "-->");

           Enumeration enumeration = s[i].getAttributes().getAll();

           while (enumeration.hasMoreElements()) {
                BasicAttribute attr = (BasicAttribute) enumeration.nextElement();
                String attrID = attr.getID();
                String attrVal = CDANameParser.unescapeAttributeValue((String)
attr.get());

                if (attrID.equals("sn")) {
                   lastName = attrVal;
```

```
                } else if (attrID.equals("givenName")) {
                   firstName = attrVal;
                } else if (attrID.equals("uid")) {
                   uidStr = attrVal;
                   uidStrEncoded = URLEncoder.encode(uidStr);
                } else if (attrID.equals("postalCode")) {
                   postalCode = attrVal;
                } else if (attrID.equals("address")) {
                   address = attrVal;
                }
             }
             theLink = details
                       + uidStrEncoded
                       + "&dn=" + dnEncoded
                       + "&auth__uid=" + uidStrEncoded
                       + "&auth__dn=" + dnEncoded;
 %>


          <tr>
            <td class="font8pt" nowrap><a href="<%= theLink %>" target="_top"><font
class="tablecontents"> <%= uidStr %></font></a></td>
<% if (isIndividual) { %>
            <td class="font8pt" nowrap><a href="<%= theLink %>" target="_top"><font
class="tablecontents"><%= firstName %></font></a></td>
<% } %>
            <td class="font8pt"><a href="<%= theLink %>" target="_top"><%= lastName
%></a></td>
            <td class="font8pt"> <%= address %></td>
          </tr>
          <% } %>
        </table>
```

```
      </td>
      <td width="3%"> </td>
    </tr>
    <tr>
      <td width="3%"> </td>
      <td colspan="3"> </td>
      <td width="3%"> </td>
    </tr>

<% }

} catch (Exception e) {
     throw e;
} finally {
       try {
              if (root != null)
                     root.remove();
       } catch(Exception e){
              // TODO: Log the exception instead of supressing it.
       }
} %>
</table>
```

Note the use of absolute paths in the inclusion, these tend to be less error prone while customizing and hence have been used. Make copies of the JSPs and then customize. Leave this set of JSPs intact because they will serve as a default search page for cases where no application specific search page is provided. Overwrite them if and only if you do not want them for even default behavior. Save the customized version in the "custom" directory, creating any sub-directory structure, as you may need.

Admin JSPs

Only a CSR with "admin" privileges will be allowed to enroll new CSR personnel and define CSR groups. The JSPs associated with enrolling CSRs are listed below:

Admin-Enrollment	Function
LeftNavAdmin.jsp	Captures navigation for Admin related tasks: namely CSR user and group search and update
AdminMainPage.jsp	Sets up a three-frame structure: top banner, left navigation, and central admin task.
AdminDefault.jsp	CSR Search Page, which includes: *searchHead.jsp*, *SearchCSRForm.jsp*, and *SearchCSRResults.jsp*.
SearchCSRForm.inc	Defines the search criteria for searching the database of enrolled CSR personnel.
SearchCSRResults.inc	Carries out the search and returns the results, creating links for adding, deleting and updating CSR profiles.
GroupDefault.jsp	The search page for CSR groups, includes JSPs*: searchHead.jsp*, *GroupSearchForm.jsp*, and *GroupSearchResults.jsp.*
GroupSearchForm.inc	Sets up search criteria to search all defined CSR groups. The product ships with the *create_schema* script that creates only the "Admin" CSR group.
GroupSearchResults.inc	Returns group search results.
GroupForm.jsp	Used to update and create new CSR group definitions.

Do not overwrite the Admin pages; only customize them as necessary.

Customizing Group Privileges

This section includes the contents of GroupForm.jsp to discuss how one may customize CSR group definition. In particular, you should note the FORM input arguments used to define a group:

- Group ID (`group__gid`)
- Description
- Privileges (`group__priv`)
- Accessible applications (`group__ddn`)

Because `group__priv` and `group__ddn` are defined using "checkboxes", to ensure that the lack of any specifications is reflected on the backend it is necessary to specify these attribute names in the hidden FORM field `definingAttribute`. But by virtue of the way HTTP FORM parameters are transferred, there are no mention of these parameters. To keep the back-end flexible enough to accept any addition custom attributes of a CSR group, this `definingAttribute` approach is used.

Typical customizations may include defining additional values for privileges and perhaps partitioning the privileges into additional subsets. In this example, they were partitioned into "Admin Access Control" and "Customer Access Control".

To assist in marking checkboxes to reflect previous values, the package com.edocs.service.util contains the class `HelpApp` and in particular the method `CheckIt(Hashtable, String)` to determine whether the specified string is in the hashtable. To use it, eaAssist places all available applications in a Hashtable and similarly all available privileges.

As always, front-end javascript can be used to enforce any business constraints that may exist. In this example, the only requirement is that a Group ID and description be specified. Needless to say, only Admin or Customer or a combination of the two is realistic.

The page itself distinguishes between the following modes:

- Add (all fields can be specified)
- Modify (all fields except group ID can be modified)
- View (no fields can be modified)

GroupForm.jsp

```
<%@ include file="searchHead.inc" %>

<SCRIPT language=javascript1.2 src="admin/scripts/groupCheck.js"></SCRIPT>

<script language="Javascript1.2">

    var msg = "<%= request.getParameter("MESSAGE") %>";

    if (msg == "null" || msg == "")

        msg = "<%= request.getAttribute("MESSAGE") %>";


    if (msg == "DUPLICATE")

        setMsg("Please enter a different Group ID. The one you selected is already in
use.");

    else if (msg == "ERROR")

        setMsg("Invalid request  or input parameters. Please check and try again.") ;

</script>

<BODY marginheight="0" marginwidth="0" bgcolor="#FFFFFF"

      onload="checkForMsg();">


<%  String task = (String) request.getParameter("task");

    boolean modify = ("modify".equalsIgnoreCase(task));

    boolean view = ("view".equalsIgnoreCase(task));

    boolean add = ("add".equalsIgnoreCase(task));

    String gidStr = (String)request.getParameter("group__gid");

    if (gidStr == null)

       gidStr = "";



    String descriptionStr  = "";

    boolean allDDNs = false;

    String[] groupDDNs = null;

    Hashtable privsHash = null;
```

```
%>

<% if (modify || view) { %>

       <%@ include file="getGroupAttrs.inc" %>

<% } else if (add) {
       // perhaps entry errors .. variant of modify ..
        gidStr = request.getParameter("group__gid");
        if (gidStr == null)
             gidStr = "";

           descriptionStr = request.getParameter("group__description");
           if (descriptionStr == null)
              descriptionStr = "";

           privsHash = HelpApp.toHash(request.getParameterValues("group__priv"));

   }
   // please note the use of the http parameter "definingAttribute"
   // give it all form parameter names that are required to describe a group
   // especially checkbox style values to ensure that if they are not selected,   //
the attribute is removed
%>

<table border="0" cellspacing="1">
  <FORM NAME=groupForm
        METHOD=POST
        TARGET="_self"
```

```
        onSubmit="return checkGroupInputs(this);"
        ACTION=CSCenter>

   <input type=hidden name=app value="CSRMain">
   <input type=hidden name=jsp value="/admin/jsp/GroupDefault.jsp">
   <input type=hidden name=returnJsp value="admin/jsp/GroupForm.jsp">
   <input type=hidden name=task value="<%= task %>">
   <input type=hidden name=nodeType value="group">
   <input type=hidden name="errforwardto" value="/admin/jsp/GroupForm.jsp">
   <input type=hidden name=definingAttribute value="gid">
   <input type=hidden name=definingAttribute value="description">
   <input type=hidden name=definingAttribute value="priv">
   <input type=hidden name=definingAttribute value="ddn">

  <tr>
    <td colspan="2" class="HeadingCells" width="984">
<%  if (add) { out.println("  Add New Group"); }
    else if (view) { out.println("  View Group"); }
    else if (modify) { out.println("  Update Group"); }
%>
   </td>
  </tr>
  <tr>
    <td width="108" class="font8ptBold"> </td>
    <td width="870" class="font8pt">
    </td>
  </tr>
  <tr>
    <td width="108" class="font8ptBold">Group ID:</td>
    <td width="870" class="font8pt">
```

```
<%  if (add) { %>
      <input type="text" id="group__gid" name="group__gid"
             size="20" maxlength="32"
              value="<%= gidStr %>"
              class="TextBoxes">
<% } else { %>
      <input type="hidden" id="group__gid" name="group__gid" value="<%= gidStr %>" >
       <%= gidStr %>
<% } %>


    </td>
  </tr>
  <tr>
    <td width="108" class="font8ptBold">Description:</td>
    <td width="870">
      <input type="text" id="group__description" name="group__description"
             size="20" maxlength="255"
             value="<%= descriptionStr  %>"
             class="TextBoxes">
    </td>
  </tr>
  <tr>
    <td width="108" class="font8ptBold"> </td>
    <td width="870">
    </td>
  </tr>
  <tr>
    <td width="978" class="font8ptBold" colspan="2" bgcolor="#CCCCCC"> Admin Access
Control:</td>
  </tr>
```

```
<tr>
  <td width="978" colspan="2">
    <table border="1" cellspacing="0" cellpadding="0">
      <tr>
        <td width="18" class="font8pt"> </td>
        <td width="150" class="font8ptBold">  Admin Tasks</td>
      </tr>
      <tr>
        <td width="18">
          <input type="checkbox" name="group__priv"
                 value="manageCSRUsers"
                <%= HelpApp.checkIt(privsHash, "manageCSRUsers")  %>
                 class="font8pt">
        </td>
        <td class="font8pt" width="150">   Manage CSR Users </td>
      </tr>
      <tr>
        <td width="18">
          <input type="checkbox" name="group__priv"
                 value="manageCSRGroups"
                <%= HelpApp.checkIt(privsHash, "manageCSRGroups")  %>
                 class="font8pt">
        </td>
        <td class="font8pt" width="150">   Manage CSR Groups </td>
      </tr>
    </table>
  </td>
</tr>
<tr>
  <td width="978" class="font8ptBold" colspan="2"> 
```

```
      </td>
  </tr>
  <tr>
    <td width="978" class="font8ptBold" colspan="2" bgcolor="#CCCCCC"> Application
Access Control:</td>
  </tr>
<tr>
    <td width="978" colspan="2">
      <table border="1" cellspacing="0" cellpadding="0">
        <tr>
          <td width="18" class="font8pt"> </td>
          <td width="150" class="font8ptBold"> Applications</td>
        </tr>
        <tr>
          <td width="18" bgcolor="#CCCCCC">
            <input type="checkbox" name="group__ddn"
                   value="*"
                  <%=  (allDDNs)?"checked" :"" %>
                   class="font8pt">
          </td>
          <td class="font8pt" width="150" bgcolor="#CCCCCC">   ALL </td>
        </tr>
<%      String[] ddns = HelpApp.getAllDDNs();

        if (ddns != null) {

           Hashtable ddnHash = HelpApp.toHash(groupDDNs);

          int num = ddns.length;
          String aDDN = null;
```

```
            for (int i=0; i < num; i++) {
               aDDN = ddns[i];


 %>
        <tr>
          <td width="18">
            <input type="checkbox" name="group__ddn"
                   value="<%= aDDN %>"
                  <%= HelpApp.checkIt(ddnHash, aDDN) %>
                   class="font8pt">
          </td>
          <td class="font8pt" width="150">  <%= aDDN %> </td>
        </tr>
<%        }
        } %>
  </table>
</tr>
  <tr>
    <td colspan="2" align="center">  </td>
  </tr>
<tr>
    <td width="978" class="font8ptBold" colspan="2" bgcolor="#CCCCCC"> Customer Access
Control:</td>
  </tr>
<tr>
    <td width="978" colspan="2">
      <table border="1" cellspacing="0" cellpadding="0">
        <tr>
          <td width="18" class="font8pt"> </td>
```

```
          <td width="150" class="font8ptBold">  Customer Tasks</td>
        </tr>

        <tr>
          <td width="18">
            <input type="checkbox" name="group__priv"
                   value="handleDispute"
                  <%= HelpApp.checkIt(privsHash, "handleDispute")  %>
                   class="font8pt">
          </td>
          <td class="font8pt" width="150">   Handle Billing Disputes </td>
        </tr>
        <tr>
          <td width="18">
            <input type="checkbox" name="group__priv"
                   value="acceptPayment"
                  <%= HelpApp.checkIt(privsHash, "acceptPayment")  %>
                   class="font8pt">
          </td>
          <td class="font8pt" width="150">   Accept Payment </td>
        </tr>
      </table>
</tr>
  <tr>
    <td colspan="2" align="center">  </td>
  </tr>
  <% if (!view) { %>
  <tr>
    <td colspan="2" align="left">
      <table border="0" cellspacing="1" width="0">
```

```
        <tr>
          <td><input type=submit name=submit value=submit class="Buttons"> </td>
          <td><input type=reset  name=reset  value=reset class="Buttons"> </td>
        </tr>
      </table>
   </td>
  </tr>
  <% } %>
  </FORM>
  <tr>
    <td colspan="2" width="984"> </td>
  </tr>
  <tr>
    <td colspan="2" width="984"> </td>
  </tr>
  <tr>
    <td colspan="2" width="984"> </td>
  </tr>
</table>
<%@ include file="/admin/html/genericFoot.htm" %>
```

About Session Timeouts

Session timeouts are set for the eaAssist application just as they are done for eStatement Manager. There is one distinction though and that stems from eaAssist behaving as a wrapper for customer pages from other applications. This occurs in the Customer Service Page. When the eStatement Manager application times out, the login information in the session is removed. After that happens, the CSR must login again into eaAssist.

It would help to select a more judicious value for the session timeout in the eStatement Manager application. The application timeout period should typically include the following: time a CSR might view the statement with interrupts from the Customer, time to perform updates to the account such as attaching notes to the customer account, and other typical “inactive” periods by the Customer or CSR during viewing of the statement.

The default session time is 600 seconds and is specified in the EJB deployment descriptor of the session bean. To modify it, one must un-jar the ejb-session.jar and then modify the ejb-jar.xml file to

either increase it or decrease it as desired. Then one must re-pack the JAR file and the EAR file that includes it, and re-deploy the application. Below are the contents of the *ejb-jar.xml* file:

```
 <?xml version="1.0" encoding="utf-8"?>

<ejb-jar xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd" version="2.1" id="ejb-jar_ID"
xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/j2ee">

       <display-name>SessionService</display-name>

       <enterprise-beans>

              <session id="Session_1">

                     <display-name>Session</display-name>

                     <ejb-name>Session</ejb-name>

                     <home>com.edocs.services.session.ISessionHome</home>

                     <remote>com.edocs.services.session.ISession</remote>

                     <ejb-class>com.edocs.services.session.Session</ejb-class>

                     <session-type>Stateful</session-type>

                     <transaction-type>Bean</transaction-type>

                     <env-entry id="EnvEntry_Session_1">

                            <env-entry-name>com.edocs.services.session.timeout</env-
entry-name>

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

                            <env-entry-value>600</env-entry-value>

                     </env-entry>

              </session>

       </enterprise-beans>

</ejb-jar>
```

There is another timeout associated with eaSuite products and it concerns data time out while composing an HTML view of a bill. This timeout is set to 900 seconds and is specified in the *web.xml* file for the application. It is a servlet parameter. Below is part of the *web.xml* file for the eaAssist application to illustrate how it is set. Note, to access the *web.xml* file the eaAssist ear must first be un-jarred, then the war within un-jarred. The *web.xml* file resides in the WEB-INF directory obtained on un-jarring the war. You must re-pack and re-deploy the application for it to take effect.

```
<?xml version="1.0" encoding="utf-8"?>

<web-app xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4"
xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
```

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/j2ee">

<display-name>eaAssist</display-name>

<distributable/>


	<servlet>

		<servlet-name>configuration-init</servlet-name>

		<servlet-class>com.edocs.fs.logging.config.api.SConfiguration</servlet-
class>

		<load-on-startup>1</load-on-startup>

	</servlet>


	<servlet>

		<servlet-name>CSRServlet</servlet-name>

		<servlet-class>com.edocs.app.AppServlet</servlet-class>

		<init-param>

			<param-name>ServletRoot</param-name>

			<param-value>com.edocs.service.app</param-value>

		</init-param>

		<init-param>

			<param-name>ErrorPage</param-name>

			<param-value>/admin/jsp/ErrorMsg.jsp</param-value>

		</init-param>

		<init-param>

			<param-name>LoginRoot</param-name>

			<param-value>com.edocs.service.app.enrollment</param-value>

		</init-param>

		<init-param>

			<param-name>LoginPage</param-name>

			<param-value>/admin/jsp/LoginCSR.jsp</param-value>

		</init-param>

		<init-param>

			<param-name>SessionTimeout</param-name>
```

```
            <param-value>900</param-value>
        </init-param>
        <init-param>
            <param-name>UserType</param-name>
            <param-value>csr</param-value>
        </init-param>
        <init-param>
            <param-name>Account.name</param-name>
            <param-value>edx/service/ejb/CSRAccount</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
        <servlet-name>CSREnrollmentServlet</servlet-name>
        <description>no description</description> -->
        <servlet-class>com.edocs.app.AppServlet</servlet-class>
        <init-param>
            <param-name>ServletRoot</param-name>
            <param-value>com.edocs.service.app.enrollment</param-value>
        </init-param>
        <init-param>
            <param-name>ErrorPage</param-name>
            <param-value>/admin/jsp/ErrorMsg.jsp</param-value>
        </init-param>
        <init-param>
            <param-name>LoginRoot</param-name>
            <param-value>com.edocs.service.app.enrollment</param-value>
        </init-param>
        <init-param>
            <param-name>LoginPage</param-name>
            <param-value>/admin/jsp/LoginCSR.jsp</param-value>
        </init-param>
```

```
        <init-param>
                <param-name>SessionTimeout</param-name>
                <param-value>900</param-value>
        </init-param>
        <init-param>
                <param-name>UserType</param-name>
                <param-value>csr</param-value>
        </init-param>
        <init-param>
                <param-name>Account.name</param-name>
                <param-value>edx/service/ejb/CSRAccount</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>CSREnrollmentServlet</servlet-name>
        <url-pattern>/CSREnrollment</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>CSRServlet</servlet-name>
        <url-pattern>/CSCenter</url-pattern>
</servlet-mapping>



<session-config>
        <session-timeout>900</session-timeout>
</session-config>
<welcome-file-list>
        <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

```
  <!-- ejb-application -->

        <ejb-ref>
              <ejb-ref-name>ejb/AppMgr</ejb-ref-name>
              <ejb-ref-type>Session</ejb-ref-type>
              <home>com.edocs.services.application.IAppMgrHome</home>
              <remote>com.edocs.services.application.IAppMgr</remote>
        </ejb-ref>
        <ejb-ref>
              <ejb-ref-name>ejb/AppIndexVolMgr</ejb-ref-name>
              <ejb-ref-type>Session</ejb-ref-type>
              <home>com.edocs.services.application.IAppIndexVolMgrHome</home>
              <remote>com.edocs.services.application.IAppIndexVolMgr</remote>
        </ejb-ref>
        <ejb-ref>
              <ejb-ref-name>ejb/AppVolMgr</ejb-ref-name>
              <ejb-ref-type>Session</ejb-ref-type>
              <home>com.edocs.services.application.IAppVolMgrHome</home>
              <remote>com.edocs.services.application.IAppVolMgr</remote>
        </ejb-ref>
        <ejb-ref>
              <ejb-ref-name>ejb/EdocsDataSource</ejb-ref-name>
              <ejb-ref-type>Session</ejb-ref-type>
              <home>com.edocs.services.application.IDataSourceHome</home>
              <remote>com.edocs.services.application.IDataSource</remote>
        </ejb-ref>

<!-- ejb-session -->
        <ejb-ref>
              <description>no description</description>
              <ejb-ref-name>ejb/Session</ejb-ref-name>
              <ejb-ref-type>Session</ejb-ref-type>
              <home>com.edocs.services.session.ISessionHome</home>
```

```
            <remote>com.edocs.services.session.ISession</remote>
        </ejb-ref>

<!-- ejb-enrollment-csr -->
        <ejb-ref>
            <ejb-ref-name>ejb/CSRAccount</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>com.edocs.enrollment.user.IAccountHome</home>
            <remote>com.edocs.enrollment.user.IAccount</remote>
        </ejb-ref>
<!-- other dependencies for application ejb -->
        <ejb-ref>
            <ejb-ref-name>ejb/VersionManager</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>com.edocs.services.versioning.IVersionManagerHome</home>
            <remote>com.edocs.services.versioning.IVersionManager</remote>
        </ejb-ref>
        <ejb-ref>
            <ejb-ref-name>ejb/VersionSetReader</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>com.edocs.services.versioning.IVersionSetReaderHome</home>
            <remote>com.edocs.services.versioning.IVersionSetReader</remote>
        </ejb-ref>
        <ejb-ref>
            <ejb-ref-name>ejb/DDFContentValidator</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>

        <home>com.edocs.services.application.validator.IContentValidatorHome</home>

        <remote>com.edocs.services.application.validator.IContentValidator</remote>
        </ejb-ref>
        <ejb-ref>
            <ejb-ref-name>ejb/BulkPublisherEJB</ejb-ref-name>
```

```
            <ejb-ref-type>Session</ejb-ref-type>

        <home>com.edocs.services.publishing.bulkpublisher.IBulkPublisherHome</home>

        <remote>com.edocs.services.publishing.bulkpublisher.IBulkPublisher</remote>
        </ejb-ref>
</web-app>
```

CSR Context

API support is provided to detect whether a CSR is viewing a bill and acting on behalf of a customer or whether it is users helping themselves. The following methods are available in the package com.edocs.service.util in the class HelpApp.

```
static public String getCSRId(HttpServletRequest req) static public Boolean
csrViewer(HttpServletRequest req)
```

The method csrViewer is particularly useful in determining whether to expose some functionality such as a button or access to information based on the viewer type. For instance, you may not want the CSR to have access to credit card information. Or perhaps you would like to update a database table column value such as "viewedOn" only if the user has viewed it. The method CSRId is useful when you would like to log who assisted a certain user.

Logging CSR Activity

eaAssist uses Java Messaging to log to the database. CSR activity is logged to the CSR_Activity table. By default, actions such as "search" for Customers/CSR/CSR-groups, insertions, deletes, updates, logins, and logouts are all logged to the CSR_Activity table. The logging happens in the java classes. For minor customization, further logging commands can be issued in the JSP pages.

Actions taken on behalf of a customer such as viewing a bill are logged to the User_Activity table with the difference that the CSR involved is recorded. Logging can be customized by making calls to the logger in the JSP pages. The process is illustrated in the CustomerDeActivate.jsp page. By default, extracting bill detail is logged to the User_Activity table.

To log activity from User applications in a conditional manner, the HelpApp methods csrViewer and getCSRId can be used. To obtain a handle to the Logger class and use it, take the following steps:

2 In your JSP, import the following classes:

```
com.edocs.service.log.ServiceLoggingConstants

com.edocs.fs.logging.*

com.edocs.fs.logging.pub.*

com.edocs.cs.util.HelpApp
```

3 Later, you can specify the following:

```
Logger.log(new UserActivityItem("", "DEACTIVATE", ddn, ddn,userId,"","",0, csrUID,
timeStr, ""));
```

The syntax for the `UserActivityItem` method is:

```
public UserActivityItem(String productCode,
                        String activityCode,
                        String ddn,
                        String app,
                        String loginId,
                        String account,
                        String billId,
                        int processedCount,
                        String flex1,
                        String flex2,
                        String createdBy)
```

The parameters for this method correlate to the following columns of the *User_Activity* table:

Column Name	Type
PRODUCT_CODE	VARCHAR2(20)
ACTIVITY_CODE	NOT NULL VARCHAR2(30)
DDN_REFERENCE	NUMBER(38)
APPLICATION	VARCHAR2(50)
LOGIN_ID	VARCHAR2(100)
ACCOUNT	VARCHAR2(100)
BILL_ID	VARCHAR2(100)
PROCESSED_COUNT	NUMBER(38)
FLEX_FIELD1	VARCHAR2(255)
FLEX_FIELD2	VARCHAR2(255)
CREATED_BY	VARCHAR2(40)

The syntax for the CSRActivityItem method is:

The syntax for the `CSRActivityItem` method is:

```
public CSRActivityItem(String productCode,
                        String activityCode,
                        String ddn,
                        String loginId,
                        String account,
                        String billId,
                        String activityStatus,
                        String flex1,
                        String flex2,
                        String comments,
                        String createdBy)
```

The parameters for this method correlate to the following columns of the *CSR_Activity* table:

Column Name	Type
PRODUCT_CODE	VARCHAR2(20)
ACTIVITY_CODE	NOT NULL VARCHAR2(30)
DDN_REFERENCE	NUMBER(38)
LOGIN_ID	VARCHAR2(100)
ACCOUNT	VARCHAR2(100)
BILL_ID	VARCHAR2(100)
ACTIVITY_STATUS	VARCHAR2(10)
FLEX_FIELD1	VARCHAR2(255)
FLEX_FIELD2	VARCHAR2(255)
COMMENTS	VARCHAR2(100)
CREATED_BY	VARCHAR2(40)

The following information is logged by eaAssist:

- CSR-Id (agent)
- Activity (CSR-Login, Logout, Search, Assist, Add, Delete, Update, or De-activate a customer)
- Activity-Status (failure, success, start, invalid session-expired, or no-account)
- Activity-sub-category (*Flex-field1* – for searches, and activity such as add/delete/update updates of customers, CSRs, or Groups)
- Customer (this is the end-user, if any, that the CSR is servicing)
- Application (DDN)

The eStatement Manager configuration files have all the information necessary for configuring the JMS logger. Refer to eStatement Manager guides on how to start and stop the JMS logger.