



Using and Customizing eaAssist™

eaAssist

V4.0
Document ID: SEGN-04-4.0-01
Data Published: 8.8.03

© 1997–2003 edocs® Inc. All rights reserved.

edocs, Inc., Two Apple Hill, 598 Worcester Road, Natick, MA 01760

The information contained in this document is the confidential and proprietary information of edocs, Inc. and is subject to change without notice.

This material is protected by U.S. and international copyright laws. edocs and eaPost are registered in the U.S. Patent and Trademark Office.

No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of edocs, Inc.

eaSuite, eaDirect, eaPay, eaCare, eaAssist, eaMarket, and eaXchange are trademarks of edocs, Inc.

All other trademark, company, and product names used herein are trademarks of their respective companies.

Printed in the USA.

Table of Contents

Preface.....	5
About Customer Self-Service and eaSuite™	5
About This Guide.....	8
Related Documentation	9
Contacting edocs Technical Support.....	10
1 Using eaAssist.....	13
Logging Into eaAssist	13
About the Banner Frame.....	15
Searching for Customer Information	15
Customer Search Results	17
Using the Main Customer Service Page	17
Defining CSR Account Information.....	19
Creating and Assigning CSR Groups	22
CSR Account Attributes	24
CSR Group Attributes	24
2 Customizing eaAssist	27
About the eaAssist EAR	27
Defining Servlets for Your Web Application	29
Defining the Context Name	31
Packaging and Deploying the Custom CSR Application	31
About the eaAssist CDA Schema.....	32
The CDA Directory Information Tree	33
Creating the CSR Enrollment Hierarchy With CDA	34
Avoiding Race Conditions.....	36
Customizing the Links to Other Applications.....	37
Custom Links	37
About the Default eaAssist JSPs.....	40
Common JSPs	40

Custom JSPs	41
Admin JSPs.....	47
Customizing Group Privileges.....	48
About Session Timeouts	55
CSR Context	58
Logging CSR Activity.....	58
Index.....	63

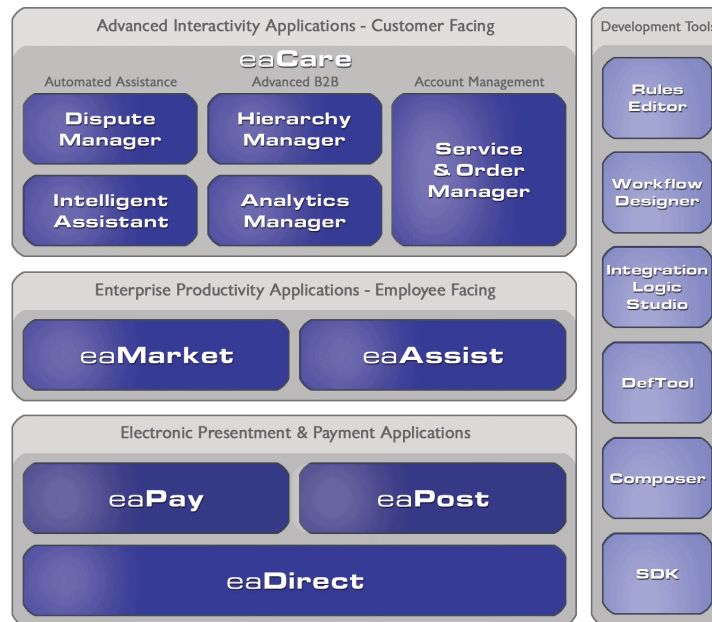
Preface

About Customer Self-Service and eaSuite™

edocs 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 is comprised 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 edocs' 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.

eaDirect™ 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 eaDirect with its data access layer, composition engine, and security, enrollment and logging framework to power complex Customer Self-Service applications.

eaPay™ is the electronic payment solution that decreases payment processing costs, accelerates receivables and improves operational efficiency. eaPay 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.

eaPost® is the account content distribution system that handles all the complexities of enrollment, authentication and secure distribution of summary account information to any endpoint, while also bringing customers back the organization's Website to manage and control their self-service experience.

Advanced Interactivity Applications are a comprehensive set of advanced customer-facing self-service capabilities that enable the full range of business and consumer customer service activities. These sophisticated modules have the flexibility to completely customize the Customer Self-Service solution to meet vertical industry and specific company requirements.

eaCare™ consists of a rich set of sophisticated self-service modules – Dispute Manager, Intelligent Assistant, Hierarchy Manager, Analytics Manager, and Service and Order Manager - for automated assistance, advanced business-to-business applications and account management. These capabilities come together to create a web self-service dashboard for customers to access all service offerings from a single, easy-to-use interface. eaCare's modularity accelerates time to market with components that can be deployed incrementally in a phased approach.

Enterprise Productivity Applications are employee-facing solutions that empower customer service representatives, sales agents, account managers, marketing managers, broker-dealers and channel partners within an organization and external partner organizations to facilitate self-service and to support assisted service. Employees leverage edocs' Customer Self-Service solution to deliver customer service, access information, create and deploy marketing and customer service content, and perform activities for the benefit of customers.

eaAssist™ reduces interaction costs and increases customer satisfaction by enabling enterprise agents – customer service representatives (CSRs), sales agents, broker-dealers and others – to efficiently access critical account data and service-related information to effectively service customers. Through its browser interface designed especially for the enterprise agent, eaAssist enables agents to take advantage of customer-facing online capabilities to provide better service by more efficiently resolving customer account inquiries at the point of customer contact.

eaMarket™ is the personalization, campaign and content management solution that enables organizations to increase revenue and improve customer satisfaction by weaving personalized marketing and customer service messages throughout the Customer Self-Service experience. The transactional account data that provides the foundation for a Customer Self-Service solution – such as transaction activity, service or usage charges, current task and prior service history – bring valuable insight into customers and can help optimize personalized marketing and customer service campaigns. eaMarket leverages that data to present relevant marketing and customer service messages to customers.

edocs' **Development Tools** are visual development environments for designing and configuring edocs' 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

This guide provides basic concepts and tools for using and customizing the eaAssist application in conjunction with the eaDirect SDK modules.

eaAssist provides a GUI interface to allow CSRs to access customer information stored in an eaDirect application. It also allows developers to write custom CSR applications against edocs applications. You will need to use the APIs described in this guide and the edocs SDK to develop and deploy such custom CSR applications.

The edocs SDK and eaAssist is intended for senior web application developers. It assumes in-depth understanding of and practical experience with:

- eaDirect 4.x 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

There are other guides included with eaAssist and the SDK documentation sets that are referenced in this guide:

Print Document	Description
<i>eaAssist Installation and Configuration Guide</i>	How to install eaAssist and configure it in a distributed environment.
<i>User Management Frameworks</i>	Describes how to build and implement a user management framework for an eaDirect deployment..
<i>Customizing and Deploying Applications</i>	Describes the basic concepts and tools for building and deploying an eaDirect J2EE application.

Contacting edocs Technical Support

Technical support is available to customers who have valid maintenance and support contracts with edocs. Technical support engineers can help you install, configure, and maintain your edocs application.

To reach the U.S. Service Center, located in Natick, MA (Monday through Friday 8:00 am to 8:00 pm EST):

- Telephone: 508.652.8400
- Toll Free: 877.336.3362
- E-support: **support.edocs.com**. This requires a one time online registration.
- E-mail: **support@edocs.com**

Whether you report a problem online, by email or by telephone, please be prepared to provide us the following information:

- What is your name and role in your organization?
- What is your company's name?
- What is your phone number and best times to call you?
- What is your e-mail address?
- In which edocs product did a problem occur?
- What is your Operating System version?
- What were you doing when the problem occurred?
- How did the system respond to the error?
- If the system generated a screen message, please send us that screen message.
- If the system wrote information to a log file, please send us that log file.

- If the system crashed or hung, please tell us.
- Do you have an idea on how to improve eaDirect? Don't hesitate to e-mail your idea to support@edocs.com.

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

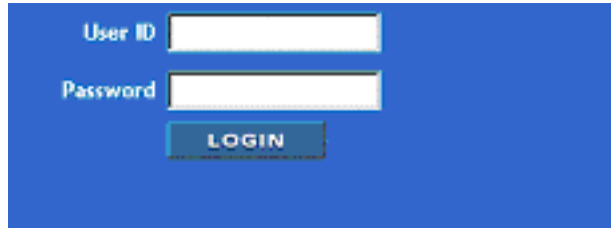
To begin using eaAssist, you must bring up the Login page by specifying the URL for eaAssist on your browser. When you installed eaAssist using the instructions in its Installation and Configuration Guide, you would have defined this URL value. It has the following syntax:

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

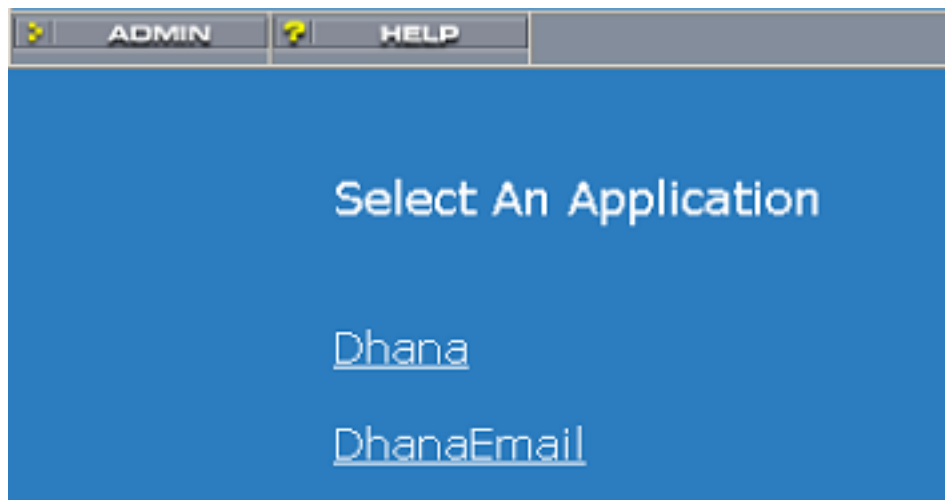
For example:

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

It brings up the Login page:

A login form on a blue background. It contains two white input fields: the first is labeled "User ID" and the second is labeled "Password". Below the "Password" field is a blue button with the text "LOGIN" in white capital letters.

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:

A screenshot of a web application interface. At the top, there is a grey navigation bar with two buttons: "ADMIN" with a yellow arrow icon and "HELP" with a yellow question mark icon. Below the navigation bar is a large blue rectangular area. In the center of this area, the text "Select An Application" is displayed in white. Below this text, there are two underlined links: "Dhana" and "DhanaEmail", both in white text.

eaAssist is initially configured with an Administration account to define CSR users and groups. By default, the User ID is **admin** and Password is **edocs** 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.



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 screenshot shows a web application window with a menu bar at the top containing 'APPLICATION', 'SEARCH', 'ADMIN', and 'HELP'. The main content area is titled 'Find Customer' and displays 'Application Name: NATW'. Below this, there are two radio buttons for 'Account Type': 'Business' and 'Individual', with 'Individual' selected. To the left of the radio buttons are two text input fields: 'Last or Business Name:' and 'Postal Code:'. To the right are two more text input fields: 'Account Number:' and 'Telephone Number:'. At the bottom left, there are three buttons: 'search', 'clear', and 'reset'. At the bottom right, there is an 'enroll' button.

The important concept to understand at this point is that eaAssist works in conjunction with an eaDirect application, and it 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 eaDirect 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 eaDirect index fields defined for an application and are expected to be customized (also part of the *LinkPages* property file). edocs recommends 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 example:

The screenshot shows a web form titled "Find Customer" with the application name "NatlWireless2". It includes fields for "Account Type" (Business and Individual), "Last or Business Name", "Postal Code", "Account Number", and "Telephone Number". Below the form is a table of search results.

User ID	First Name	Last Name	Address
User 1432595	edocs	edocs	1 main st

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.

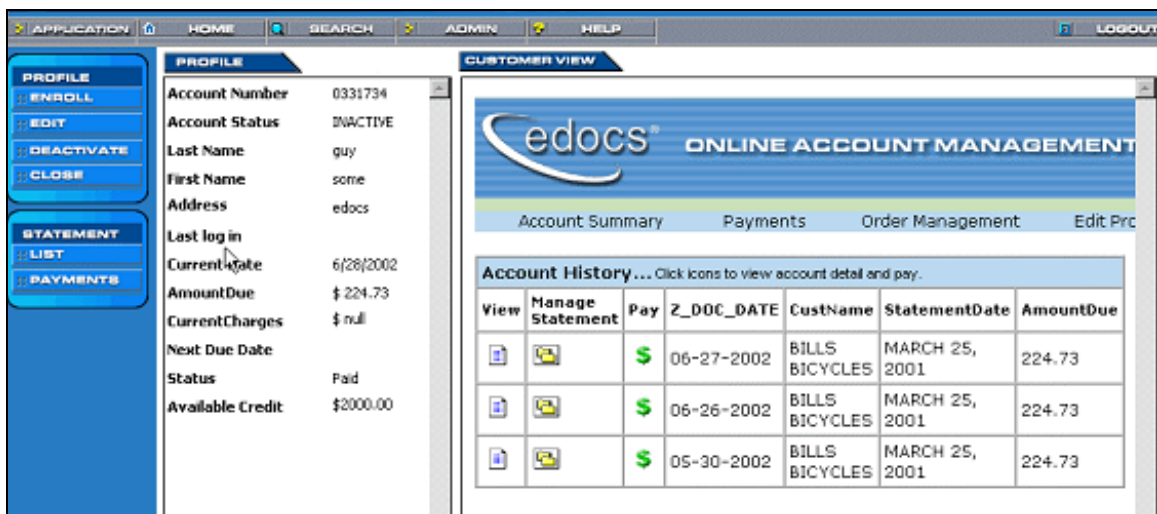
Using the Main Customer Service Page

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:

Using eaAssist

- 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 eaDirect and available to the customer and CSR.
- **Payment:** if your application also has eaPay 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 eaDirect application. All of this information is stored in the eaDirect 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 eaDirect 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:

Using eaAssist

Search CSR User

User ID: Group ID:

First Name: Last Name:

Registered Users. Select User to edit.

User ID	Group ID	First Name	Last Name	Description	Delete
admin	Admin	Admin	Powerful	CSR Super User	
andrea	someDONS	andrea	andrea	andrea	<input type="button" value="X"/>
malini	someDONS	Malini	Bhandaru	asjndfjk	<input type="button" value="X"/>

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 named edocs 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:

Update CSR

Personal Information

First Name :

Last Name : Password :

User ID : Confirm Password :

Description :

E-mail :

Assign to group:

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:

Add New CSR

Personal Information

First Name :

Last Name : Password :

User ID : Confirm Password :

Description :

E-mail :

Assign to group:

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:

Update Group

Group ID: Admin

Description:

Admin Access Control:

	Admin Tasks
<input checked="" type="checkbox"/>	Manage CSR Users
<input checked="" type="checkbox"/>	Manage CSR Groups

Application Access Control:

	Applications
<input checked="" type="checkbox"/>	ALL
<input checked="" type="checkbox"/>	One
<input checked="" type="checkbox"/>	Test
<input checked="" type="checkbox"/>	test
<input checked="" type="checkbox"/>	two

Customer Access Control:

	Customer Tasks
<input type="checkbox"/>	Handle Billing Disputes
<input type="checkbox"/>	Accept Payment

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	<i>givenName</i>	Text/32
Last Name	<i>sn</i>	Text/64
Password	<i>userPassword</i>	Text/15
Description	<i>description</i>	Text/255
Email	<i>mail</i>	Text/127
Group	<i>group</i>	Pre-defined

CSR Group Attributes

The following table describes the attributes of a CSR Group.

Description	CDA Attribute	Type/Length
Group Id	<i>gid</i>	Text/32

Description	<i>description</i>	Text/64
Privilege	<i>priv</i>	Multiple values, text, possibly none
Data Definition Name	<i>ddn</i>	Multiple values, text, possibly none

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 6.1 J2EE platform with Service Pack 2. 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 (Enterprise Application aRchive). When you install and configure eaAssist along with eaDirect, it deploys the following EAR files to the `<weblogic-home>/config/mydomain/applications` directory, where `<weblogic_home>` is the directory location where you installed WebLogic:

Application Files	Description
<i>ear-eadirect.ear</i>	Command Center application
<i>ear-easample.ear</i>	eaSample user application
<i>ear-eatraining.ear</i>	Custom hierarchical application
<i>ear-ebills.ear</i>	Old enrollment model application
<i>ear-eapay.ear</i> <i>ear-eapost.ear</i> <i>ear-eaxchange.ear</i>	Skeleton auxiliary applications
<i>ear-eaassist.ear</i>	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-eaassist.ear*) from `<eaAssist_home>/J2EEApps/weblogic` to your working directory, where `<eaAssist_home>` is the location where you installed eaAssist.
3. Extract the files out of the EAR file by using the following command from your working directory:

```
jar xvf ear-eaassist.ear
```

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

Delete the *ear-eaassist.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-eaassist.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-eaassist.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*
- The eaAssist JSPs, Java scripts, and HTML pages located in the *admin* directory.
- The files *create_schema* and *index.html*.

create_schema defines the CSR tables in the existing eaDirect CDA database. This schema change is defined later in this chapter. 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.

For example, the eaDirect UserServlet and UserEnrollment servlets both use the *AppServlet.class* file located in *WEB-INF/classes/com/edocs/app*. If you want to create a new servlet to access a class file similar to *AppServlet.class*, 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*.
2. 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).
3. Edit *WEB-INF/weblogic.xml* to change all instances of *eaAssist* to *newcsrapp*. Do the same for the *web.xml* file.

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:

1. 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 eaDirect guide *Customizing and Deploying Applications*.

About the eaAssist CDA Schema

eaAssist extends the default edocs 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 edocs user management framework and how to customize it, you should consult the guide: *User Management Frameworks*.

The CDA Directory Information Tree

The edocs 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 is run during installation. The *create_schema* file is used as input to a java process that alters the eaDirect CDA database for enrollment information. For information about how to run that CDA script, see the eaAssist *Installation and Configuration Guide* for your platform.

CSR enrollment and privilege information is stored on a separate branch of the CDA database of eaDirect. 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/end-user 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
"*"

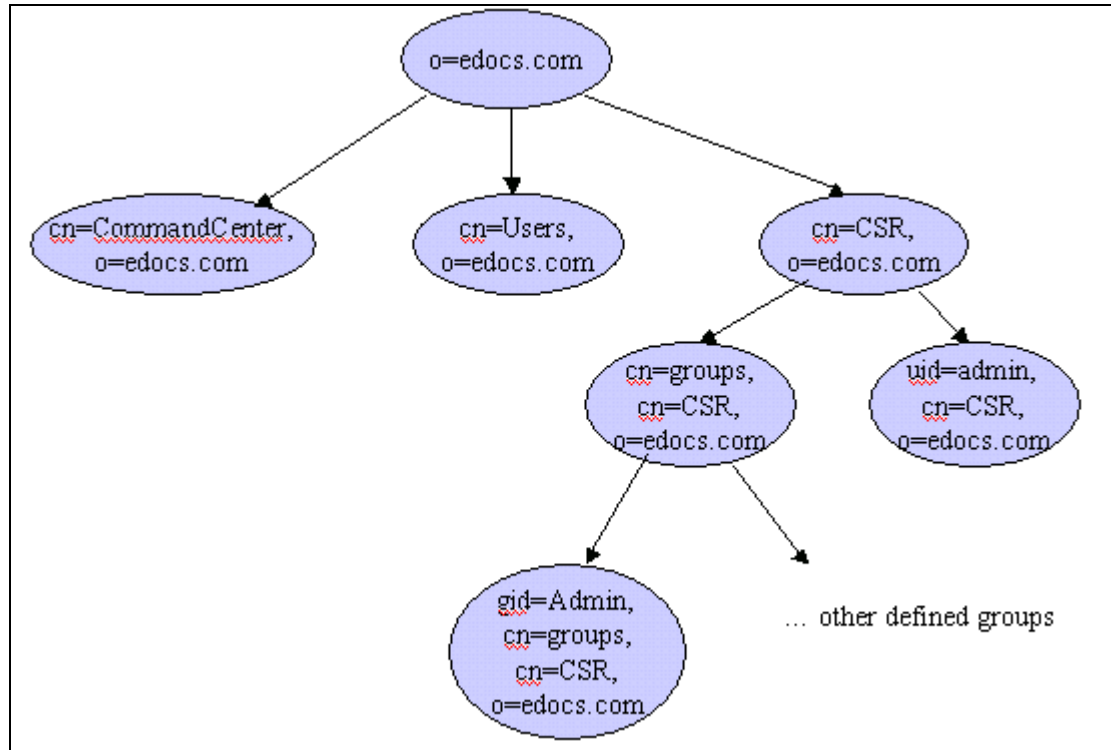
mk uid=admin,cn=CSR,o=edocs.com userPassword D2B71E9C2E21C5F2 uid
admin givenName Admin sn Powerful mail admin@edocs.com group
"gid=Admin,cn=Groups,cn=CSR,o=edocs.com" description "CSR Super
User"

```

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.

The following shows the eaDirect CDA database after the eaAssist CSR related nodes are created (using the `mk` command):



Avoiding Race Conditions

The edocs 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/cs/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/eaSample/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=/eaSample/UserEnrollment?app=SubscribeApp&jsp=
/enrollment/jsp/user_get_subscribe.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_subscribe.jsp
```

The same thing applies for the pay, summary, edit, 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/eaSample/ejb/CDAAccount
deactivate	/eaAssist/CSCenter?app=CSRMain&submit=deleteAcct&jsp=/admin/jsp/CustomMainPage.jsp
summary	/eaSample/User?app=UserMain&jsp=/user/jsp/HistoryList.jsp
detail	/eaSample/User?app=UserMain&jsp=/user/jsp/Detail.jsp
edit	/eaSample/UserEnrollment?app=UpdateApp&jsp=/enrollment/jsp/user_get_subscribe.jsp
enroll	/eaSample/UserEnrollment?app=SubscribeApp&jsp=/enrollment/jsp/user_get_subscribe.jsp
pay	/eaAssist/CSCenter?app=CSRMain&jsp=/admin/html/PayHistory.htm
profile	/admin/jsp/CustomProfile.jsp
search	/admin/jsp/CustomSearch.jsp
summary	/eaSample/User?app=UserMain&jsp=/user/jsp/HistoryList.jsp

Note how some of the links go into a separate web application, in this case eaSample 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 eaSample 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. Please use then 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
<i>customer.jsp</i>	Defines DDN and uid variables.
<i>nav.jsp</i>	Imports <i>LinkPages.java</i> to facilitate custom page links.
<i>privs.jsp</i>	Obtains the CSR's privileges and makes them accessible to the JSP pages for controlling access to various functionality.
<i>searchHead.jsp</i>	Java class imports necessary for a CDA-based search.
<i>ErrorMsg.jsp</i>	Used to return a standard error message, whose source contains details about the error message as a hidden field.
<i>UserMsg.jsp</i>	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, please 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 eaDirect deployment more closely or perhaps have variants for each customer application running against a particular eaDirect 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 "edocs__" 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
<i>CustomerMainPage.jsp</i>	Four frame page setup: banner, left navigation, customer profile and customer detail
<i>CustomerProfile.jsp</i>	Customer profile page, typically is customized based on the enrollment data available and the indexed fields in the application.
<i>CustomerSearch.jsp</i>	Search criteria and jsp inclusions for search and their results.
<i>CustomerSearchForm.inc</i>	Form for specifying search criteria. Note the <code>user__</code> prefix for form fields that are required to be propagated.
<i>CustomerSearchResults.inc</i>	Search result display for the sample billing application, includes conditional logic based on account type.
<i>CustomerDeActivate.jsp</i>	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.
<i>LeftNavigation.jsp</i>	This page is customized indirectly via the <code>LinkPages</code> class and <i>csrLinks.properties</i> file. Its goal is to provide the CSR navigation capabilities on the main customer page.
<i>Banner.jsp</i>	Provides navigation ability across eaAssist. You may need to customize it for online Help.
<i>OnePage.jsp</i>	Has in-line frames that display <code>CustomerProfile</code> and <code>Customer View</code> pages. The <code>Customer View</code> 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.jsp" %>

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

<table width="100%" border="0">
  <tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
  </tr>
</table>
<center>

<% Properties searchEnv = HelpApp.getParamsNoTrim(request,
"user__");
String submit = request.getParameter("submit");
%>
<%@ include file="/admin/jsp/CustomerSearchForm.jsp" %>
<%
  if ((submit != null) && submit.equals("search")) { %>
    <%@ include file="CustomerSearchResults.jsp" %>
  <% } %>

</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 that the fields that will be necessary to authenticate the users towards the CSR serving them must be specified using “auth__” as a prefix. So, in the example account implementation 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`”.

Please 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));
        }
    }
```



```
<%
    // using last name to sort
    s = HelpApp.sortSearchResults(s, "sn");
    for (int i = 0; i<s.length; ++i) {
        String uidStr = null;
        String firstName= null;
        String lastName = null;
        String address = null;
        String postalCode = null;
        String theLink = null;
        Enumeration enum = s[i].getAttributes().getAll();
        while (enum.hasMoreElements()) {
            BasicAttribute attr = (BasicAttribute) enum.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;
            } else if (attrID.equals("postalCode")) {
                postalCode = attrVal;
            } else if (attrID.equals("address")) {
                address = attrVal;
            }
        }
        theLink = details + URLEncoder.encode( uidStr)
            + "&auth__uid=" + URLEncoder.encode( uidStr)
            + "&auth__dn=" + URLEncoder.encode(HelpApp.
            getCustomerRDN(ddn, uidStr));
    }
%>
```

```

        <tr>
            <td class="font8pt" nowrap><a href="<%= theLink %>" target="_top">
                <font class="tablecontents">&nbsp;<%= 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">&nbsp;<%= address %></td>
        </tr>
        <% } %>
    </table>
</td>
<td width="3%">&nbsp;</td>
</tr>
<tr>
    <td width="3%">&nbsp;</td>
    <td colspan="3">&nbsp;</td>
    <td width="3%">&nbsp;</td>
</tr>
<% }
} catch (Exception e) {
    throw e;
} finally {
    if (root != null) root.remove();
} %>
</table>

```

Please note the use of absolute paths in the inclusion, these tend to be less error prone while customizing and hence have been used. Please make copies of these JSPs and 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
<i>LeftNavAdmin.jsp</i>	Captures navigation for Admin related tasks: namely CSR user and group search and update.
<i>AdminMainPage.jsp</i>	Sets up a three-frame structure: top banner, left navigation, and central admin task.
<i>AdminDefault.jsp</i>	CSR Search Page which includes: <i>searchHead.jsp</i> , <i>SearchCSRForm.jsp</i> , and <i>SearchCSRResults.jsp</i> .
<i>SearchCSRForm.inc</i>	Defines the search criteria for searching the database of enrolled CSR personnel.
<i>SearchCSRResults.inc</i>	Carries out the search and returns the results, creating links for adding, deleting and updating CSR profiles.
<i>GroupDefault.jsp</i>	The search page for CSR groups, includes JSPs: <i>searchHead.jsp</i> , <i>GroupSearchForm.jsp</i> , and <i>GroupSearchResults.jsp</i> .
<i>GroupSearchForm.inc</i>	Sets up search criteria to search all defined CSR groups. The product ships with the <i>create_schema</i> script that creates only the “Admin” CSR group.
<i>GroupSearchResults.inc</i>	Returns group search results.
<i>GroupForm.jsp</i>	Used to update and create new CSR group definitions.

Please do not overwrite these Admin pages; only customize these 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.
src="admin/scripts/groupCheck.js"></SCRIPT>
<BODY marginheight="0" marginwidth="0" bgcolor="#FFFFFF">

<% 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");
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");
    descriptionStr = request.getParameter("group__description");
    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=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("&nbsp; Add New Group"); }
  else if (view) { out.println("&nbsp; View Group"); }
  else if (modify) { out.println("&nbsp; Update Group"); }
%>
  </td>
</tr>
<tr>
  <td width="108" class="font8ptBold">&nbsp;</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="64"
      value="<%= descriptionStr %>"
      class="TextBoxes">
    </td>
</tr>
<tr>
  <td width="108" class="font8ptBold">&nbsp;</td>
  <td width="870">
    </td>
</tr>
<tr>
  <td width="978" class="font8ptBold" colspan="2" bgcolor="#CCCCC">
    Admin Access Control:</td>
</tr>

```

```

<tr>
  <td width="978" colspan="2">
    <table border="1" cellspacing="0" cellpadding="0">
      <tr>
        <td width="18" class="font8pt">&nbsp;</td>
        <td width="150" class="font8ptBold">&nbsp;  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"> &nbsp;  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"> &nbsp;  Manage CSR Groups </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr>
      <td width="978" class="font8ptBold" colspan="2">&nbsp;  </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">&nbsp;</td>
        <td width="150" class="font8ptBold">&nbsp;<Application></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"> &nbsp;<ALL
          </td>
        </tr>
      </table>
      <%=
        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"> &nbsp;<%= aDDN %> </td>
          </tr>
        <%=
        }
      <%>
    </table>
  </tr>
  <tr>
    <td colspan="2" align="center">&nbsp;</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">&nbsp;</td>
        <td width="150" class="font8ptBold">&nbsp;< 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"> &nbsp;< 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"> &nbsp;< Accept Payment </td>
          </tr>
        </table>
      </tr>
      <tr>
        <td colspan="2" align="center">&nbsp;</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 eaDirect. 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 eaDirect 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 eaDirect 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 unjar 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"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
Enterprise JavaBeans 1.1//EN" 'http://java.sun.com/j2ee/dtds/ejb-
jar_1_1.dtd'>

```

```

<ejb-jar>
  <display-name>SessionService</display-name>
  <enterprise-beans>
    <session>
      <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>
        <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.


```

<web-app>
  <display-name>eaAssist</display-name>
  <description>edocs Customer Service application</description>
  <distributable></distributable>
  <servlet>
    <servlet-name>CSRServlet</servlet-name>
    <display-name>CSRServlet</display-name>
    <description>no description</description>
    <servlet-class>com.edocs.app.AppServlet</servlet-class>
    <init-param>
      <param-name>ServletRoot</param-name>
      <param-value>com.edocs.cs.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.cs.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>Account.name</param-name>
      <param-value>java:comp/env/ejb/CSRAccount</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

```

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:

1. 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
```

2. 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
<i>PRODUCT_CODE</i>	VARCHAR2(20)
<i>ACTIVITY_CODE</i>	NOT NULL VARCHAR2(30)
<i>DDN_REFERENCE</i>	NUMBER(38)
<i>APPLICATION</i>	VARCHAR2(50)
<i>LOGIN_ID</i>	VARCHAR2(100)
<i>ACCOUNT</i>	VARCHAR2(100)
<i>BILL_ID</i>	VARCHAR2(100)
<i>PROCESSED_COUNT</i>	NUMBER(38)
<i>FLEX_FIELD1</i>	VARCHAR2(255)

<i>FLEX_FIELD2</i>	VARCHAR2(255)
<i>CREATED_BY</i>	VARCHAR2(40)

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
<i>PRODUCT_CODE</i>	VARCHAR2(20)
<i>ACTIVITY_CODE</i>	NOT NULL VARCHAR2(30)
<i>DDN_REFERENCE</i>	NUMBER(38)
<i>LOGIN_ID</i>	VARCHAR2(100)
<i>ACCOUNT</i>	VARCHAR2(100)
<i>BILL_ID</i>	VARCHAR2(100)
<i>ACTIVITY_STATUS</i>	VARCHAR2(10)
<i>FLEX_FIELD1</i>	VARCHAR2(255)
<i>FLEX_FIELD2</i>	VARCHAR2(255)
<i>COMMENTS</i>	VARCHAR2(100)
<i>CREATED_BY</i>	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 eaDirect configuration files have all the information necessary for configuring the JMS logger. Please refer to eaDirect guides on how to start and stop the JMS logger.

Index

A

- Add New CSR page, 21
- Admin JSPs, 47
- Admin user, 20
- Application Selection page, 14
- Authentication, 36

B

- Banner, 15
- Bind attributes, 33

C

- CDA schema, 32
- Class files, 30
- Common JSPs, 40
- Context name, 31
- Creating CSR groups, 22
- CSR Account Information, 19
- Custom JSPs, 41
- Customer profile, 19
- Customer Search page, 15
- Customer Self-Service, 5
- Customizing Links, 37

D

- DDN, 14
- Deactivate, 19
- Default links, 38
- Delete CSR group, 22
- Deploy CSR Application, 32
- Deployment descriptors, 29
- Directory information tree, 33

E

- EAR, 16, 27
- eaSuite, 5
- Enrollment, 16
- Enrollment hierarchy, 34

G

- Groups, 22

L

- LinkPages, 37
- LinkPages file, 16
- Logging, 58
- Login page, 13

M

Main CSR page, 17

P

Packaging CSR
application, 31

Password, 14

Payment, 19

R

Race conditions, 36

S

Schema, 32

Schema names, 16

SDK modules, 8

Search CSR group, 22

Search fields, 16

Search results, 17

Servlets, 28, 29

Session timeouts, 55

T

Technical support, 10

Timeouts, 55

U

Update CSR page, 20

User ID, 14

W

WAR file, 28

Wild card, 16