Peregrine Systems, Inc.

AssetCenter Web 4.4

Administration Guide

Peregrine
S Y S T E M S®

Contents

PEREGRINE

About this Guide

This guide includes the following topics:

- Performing administrative tasks in AssetCenter Web
- Configuring AssetCenter Web for AssetCenter
- Understanding how users are identified in AssetCenter Web
- Using the Peregrine Portal
- Personalizing forms

Book audience

This guide is intended for administrators who configure and maintain AssetCenter Web. To use this guide effectively, you must have a working knowledge of the following:

- Operating guides, reference manuals, and other documentation for your PC hardware and operating system
- AssetCenter administration and functionality
- XML knowledge

Related documentation

Refer to the following documentation for additional information:

Documentation	Description
AssetCenter Web Installation Guide	This describes how to install and configure the Peregrine OAA platform, AssetCenter Web, Java SDK, and application and Web servers.
AssetCenter Web Tailoring Kit Guide	This describes how to use the AssetCenter Web tailoring kit. Topics covered in this guide include: installation of the tailoring kit, a list of common tailoring tasks, how to use Peregrine Studio, how to tailor forms and form components, how to use scripting, and troubleshooting information.

Associated applications

This guide does not contain information about products that you use with AssetCenter Web, such as AssetCenter. Refer to the appropriate product documentation for information about installing, configuring, and using associated applications.

Note: You must install and configure AssetCenter before you can install and configure AssetCenter Web. Refer to the AssetCenter Web Installation Guide and AssetCenter Web Administration Guide for instructions.

Terminology

The terminology used in this guide and in the AssetCenter Web interface is based on AssetCenter 3.6 and 4.x.

Typographical conventions

This guide uses typeface conventions to indicate special terms and actions. These conventions and their meanings are:

Convention	Meaning
Bold	Information that you must type exactly as shown appears in bold. The names of buttons, menus, and menu options also appear in bold.
Italics	Variables and values that you must provide appear in *italics*. New terms also appear in *italics*.

Monospace	Code or script examples, output, and system messages appear in a monospace font.
	`var msgTicket = new Message( "Problem" );`
	…
	`msgTicket.set( "_event", "epmc" );`
	
	An ellipsis (…) indicates that portions of a script are omitted because they are not needed for the current topic. Samples of code are not entire files, but they are representative of the information discussed in a particular section.
	Filenames, such as `login.asp`, appear in a monospaced font.

Special elements

This book uses special elements to help you locate information. These special elements and their uses are in the following table:

Element	Usage
Important:	Information that is required to complete a task
Note:	Information that is of general interest
Tip:	Information that can make a task easier or faster
Warning:	Information that is needed when there is a risk of losing data

Organization of the guide

The following table shows you where in this guide to find the information you need.

To find this …	Look here…
Peregrine OAA architecture overview	Chapter 1, Architecture Overview
AssetCenter Web features and architecture overview	Chapter 2, AssetCenter Web Overview
Customizing the AssetCenter Web interface	Chapter 2, Customizing the Peregrine Portal
Configuring and using the Peregrine Portal	Chapter 3, Using the Peregrine Portal
Activating and using end user interface personalization	Chapter 4, Using the Personalization Interface
Using document schema definitions and schema extensions	Chapter 5, Document Schema Definitions
Customizing forms that cannot be personalized from the browser	Chapter 6, Modifying the Request Type and Item Type Selection Menus
Tree menu enhancements	Chapter 7, Tree Menu Enhancements
Administering AssetCenter Web using the Admin module	Chapter 8, AssetCenter Web Administration
Configuring AssetCenter Web to work with AssetCenter	Chapter 9, Back-end System Administration
Security features	Chapter 10, Security
Troubleshooting issues	Chapter 11, Troubleshooting

Need further assistance?

For further information and assistance with this release, you can download documentation or schedule training.

Customer Support

For further information and assistance, contact Peregrine Systems' Customer Support at the Peregrine CenterPoint Web site.

To contact customer support:

1 In a browser, navigate to *http://support.peregrine.com*

2 Log in with your user name and password.

3 Follow the directions on the site to find your answer. The first place to search is the KnowledgeBase, which contains informational articles about all categories of Peregrine products.

4 If the KnowledgeBase does not contain an article that addresses your concerns, you can search for information by product; search discussion forums; and search for product downloads.

Documentation Web site

For a complete listing of current AssetCenter Web documentation, see the Documentation pages on the Peregrine Customer Support Web.

To view the document listing:

1 In a browser, navigate to *http://support.peregrine.com*.

2 Log in with your login user name and password.

3 Click either Documentation or Release Notes at the top of the page.

4 Click the AssetCenter Web link.

5   Click a product version link to display a list of documents that are available for that version of AssetCenter Web.

6   Documents may be available in multiple languages. Click the Download button to download the PDF file in the language you prefer.

You can view PDF files using Acrobat Reader, which is available on the Customer Support Web site and through Adobe at *http://www.adobe.com*.

Important: Release Notes for this product are continually updated after each release of the product. Ensure that you have the most current version of the Release Notes.

Education Services Web site

Peregrine Systems offers classroom training anywhere in the world, as well as "at-your-desk" training using the Internet. For a complete listing of Peregrine's training courses, refer to the following web site:

http://www.peregrine.com/education

You can also call Peregrine Education Services at +1 858.794.5009.

1

CHAPTER

Architecture Overview

Peregrine Open Application Architecture (OAA) is a software platform that enables the hosting of a variety of Web applications over a corporate intranet. The platform is Java based, encompassing the latest in Java technology including Java servlets, JAAS login authentication, and JSP pages that enable Web pages to display data dynamically.

Peregrine OAA overview

Peregrine OAA is the underlying architecture for many Peregrine products, including, but not limited to, the Get-It suite of Employee Self-Service products.

OAA Product	Description
AssetCenter Web	Web-based application that enables access to the AssetCenter database to all users without having to install the AssetCenter client.
BI Portal	Web-based reporting tool for creating and executing queries against ServiceCenter 5.1 data; and for generating reports and graphs based on that data.
Get-Answers	Web-based, knowledge management application that enables you to capture and store knowledge in a database, and to search for that knowledge when you need it. Using Get-Answers, you can improve the quality and accuracy of the knowledge that people in your company use to perform their jobs, and help them avoid calls to the service desk.

OAA Product	Description
Get-Resources™	Web-based solution that integrates with AssetCenter Procurement, AssetCenter Portfolio, or ServiceCenter Request Management to enable employees to create requests for resources and to streamline the approval workflow of those requests throughout the organization.
Get-Services™	Web-based extension of ServiceCenter that enables users to report problems in the work environment by opening problem tickets in Get-Services and then store them in the ServiceCenter back-end system. This allows users to view tickets from Get-Services and ServiceCenter. Modules include Service Desk and Change Management.

Peregrine OAA provides a Web portal, Peregrine Portal, from which users can access their Web applications. The Peregrine Portal also provides access to the Admin module, from which all aspects of Peregrine OAA are monitored and maintained.

The base of Peregrine OAA includes:

Component	Description
Archway	A Java servlet that processes HTTP requests from a browser, sends the requests through an adapter to a back-end system, and returns XML data to be displayed in the browser.
Core files	Peregrine OAA contains jsp and XML. The core consist mainly of low level Java utility classes used by the Portal Web applications built on the base OAA framework.
Peregrine Portal	Includes a login page and provides access to your Peregrine Web applications and to the Admin module for configuration of your application.
Skins and style sheets	Provide a choice for the appearance of the Web pages.

Peregrine OAA includes a number of components that are configured for use with Web applications as they are needed. These include:

Component	Description
Adapters	Enables connection to the back-end system database. The adapter required by your Web application is deployed during the installation.
OAA Persistence (Get-Answers only)	Provides a general purpose database that is used by certain Peregrine Web applications. OAA Persistence provides data persistence to a database.

Component	Description
OAA Workflow (Get-Answers only)	Enables workflow capabilities used by some Peregrine OAA Web applications.
Notification Services (Get-Answers only)	A centralized service for sending and receiving notifications through multiple communication devices and for tracking the status of these notifications.

Separate documentation for Notification Services is provided with the Web applications that use this feature.

Peregrine OAA architecture

Peregrine OAA applications and interfaces use Web-based building blocks that include:

HTTP	A simple and widely supported protocol for sending client requests to a server. Variations such as HTTPS provide security as well.
XML	Extensible Markup Language. A documentation meta-language that allows you to format data, which can then be displayed through a Web browser. Unlike HTML, you create your own XML tags and define them any way you want.
Commercial Web servers	The services provided by the Archway architecture can be served from any commercial Web server, including IIS and Apache.
Application servers	Peregrine OAA supports Apache Tomcat, WebSphere, and WebLogic.
Common clients	Applications can be deployed with Web browsers (for example, IE, Netscape, and Mozilla), handheld devices (Palm Pilot), or mobile phones (through HDML).

The application server processes data (JSP pages, XML, and so forth) that it receives from the database or client that is specifically related to the Peregrine Systems Web applications. The Web server converts the data into a form (HTML) that can be displayed in a Web browser.

The following diagram illustrates the architecture:
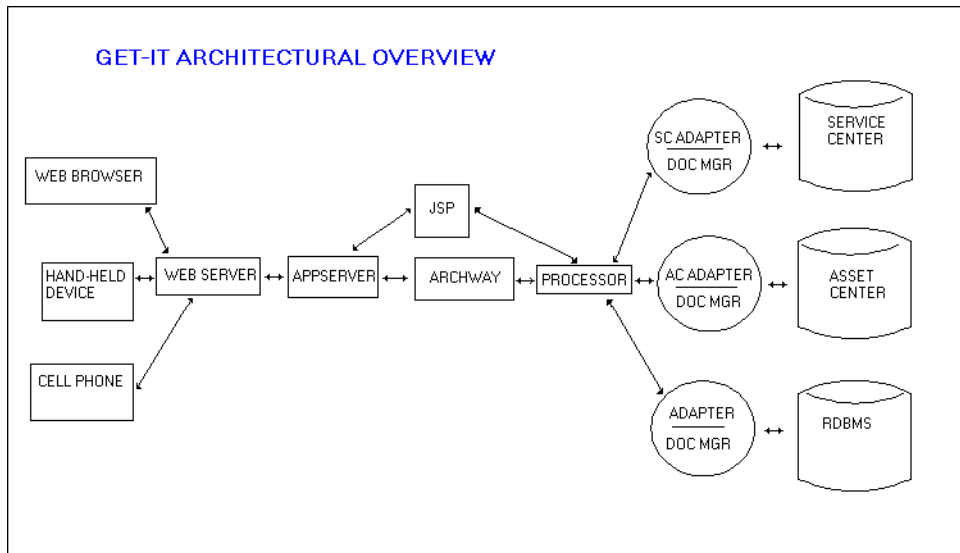
The Archway component listens to HTTP requests from clients, routes the requests to an appropriate server, and returns data or documents. The requests supported by Archway can vary, but they fundamentally consist of queries, data updates, or system events.

For example, a client can contact Archway and ask to query a database for a list of problem tickets. Another client can contact Archway and supply it with a new purchase request to be entered into the database.

All requests and responses are formatted using XML, such as the following problem ticket expressed in XML.

```
<problem>
   <number>PM5670</number>
   <contact> Joe Smith </contact>
   <description> My printer is out of paper </description>
</problem>
```

Clients that interact with Archway can do anything they need with the XML that is returned as a response. Very frequently, the client initiating the request is a user interface such as a Web browser. Such a client could easily display the XML documents returned by Archway. However, to be of better use, the XML

documents are often displayed within a formatted HTML page. This is accomplished by using Java Server Pages (JSP).

JSP provides a syntax for creating HTML pages that is pre-processed by the Web server before being sent to the browser. During this processing, XML data obtained from Archway is merged into the HTML page.

Archway's architecture includes special support for automatically generating the HTML and JSP pages that make up a Web application.

OAA scalability

You can ensure that OAA applications perform well as the number of users in your organizations grows. For complete information, see the Guide to OAA architecture and optimization, which is available for download in PDF format in the Employee Self Service section of Product News at *http://support.peregrine.com/support/Get-Resources*.

Archway internal architecture

Archway is implemented as a Java servlet. The Java servlet is an application executed by a Web server that processes HTTP requests from client Web browsers and sends the request, by way of an adapter, to a database. It then retrieves the requested information from the database and returns it to the client. Archway requires both a Java environment and a Web server.

Each request is interpreted to determine its destination. Archway is able to communicate with a variety of back-end systems, including the AssetCenter or ServiceCenter products from Peregrine.

Requests can be handled in one of three ways:

- A request can be sent directly to an adapter that talks to a back-end server. For instance, a query request for opened tickets could be forwarded to an adapter capable of communicating with AssetCenter.
- A request can be sent to a script interpreter hosted by Archway. This enables you to define your own application-specific services. Within a script, calls

can be made back to Archway to access the back-end system with database operations and events.

■ Finally, a request can be sent to a component known as a Document Manager. This component provides automated services for combining logical documents.

Archway communicates with back-end systems with the help of specialized adapters that support a predefined set of interfaces for performing connections, database operations, events, and authentication. All adapters use DLLs to communicate with each application.

Messages can be routed to a script interpreter hosted by Archway. The interpreter supports ECMAScript, a European standard based on the Core JavaScript language used by Netscape (JavaScript) and Microsoft Internet Explorer (JScript).

Messages can be routed to the Document Manager component. This component reads special schema definitions that describe application documents for logical entities such as a purchase request, problem ticket, or product catalog. The script interpreter uses these schemas to automatically generate database operations that query, insert, or update such documents.

Archway requests

Archway supports a variety of requests, all of which are based on two basic technologies: HTTP and XML. The HTTP protocol defines a simple way for clients to request data from a server. The requests are stateless and a client/server connection is maintained only during the duration of the request. All this brings several advantages to Archway, including the ability to support a large number of requests with the help of any of today's commercial Web servers.

Another important advantage is that any system capable of making HTTP requests can contact Archway. This includes Web browsers, of course. But in addition, all modern programming environments support HTTP. This makes it very simple to write new adapters that communicate with Peregrine servers without the need of specialized APIs.

You can test the output generated by your server-side onload scripts and schemas by using URL queries to the Archway servlet.

Archway will invoke the server script or schema as an administrative user and return the output as an XML document. Your browser will need an XML renderer to display the output of the XML message.

Note: Your browser may prompt you to save the XML output of the URL query to an external file.

URL Script Queries

Archway URL script queries use the following format:

```
http://server name/oaa/servlet/archway?script name.function name
```

- For `server name`, enter the name of the Java-enabled Web server. If you are testing the script from the computer running the Web server, you can use the variable `localhost` as the server name.

   The `/oaa/servlet` mapping assumes that you are using the default URL mapping that AssetCenter Webautomatically defines for the Archway servlet. If you have defined another URL mapping, replace the servlet mapping with the appropriate mapping name.

- For `script name`, enter the name of the script you want to run.
- For `function name`, enter the name of the function used by the script.

Note: URL queries functionality can be removed by configuring the `WEB.xml` file. This is a recommended security setting.

URL Schema Queries

Archway URL schema queries use the following format:

```
http://server name/oaa/servlet/archway?adapter name.Querydoc
&_document=schema name
```

- For `adapter name`, enter the name for the back-end database adapter the schema uses. The adapter listed here will use the ODBC connection that you have defined in the Admin module Settings page.
- For `schema name`, enter the name defined in the `<document name="schema name">` element of the schema file.

   The `/oaa/servlet` mapping assumes that you are using the default URL mapping that AssetCenter Web automatically defines for the Archway

servlet. If you have defined another URL mapping, replace the servlet mapping with the appropriate mapping name.

URL SQL Queries

Archway URL SQL queries use the following format:

```
http://server name/oaa/servlet/archway?adapter name.query&_table=
table name&field name=value&_[optional]=value
```

- For `adapter name`, enter the name for the back-end database adapter the schema uses. The adapter listed here will use the ODBC connection that you have defined in the Admin module Settings page.

- For `table name`, enter the SQL name of the table you want to query from the back-end database.

- For `field name`, enter the SQL name of the field you want to query from the back-end database.

- For `value`, enter the value you want to the field or optional parameter to have.

- For `_[optional]`, enter any optional parameters to limit your query. Examples include:

  - `_return`. Returns the values only of the fields you list.

  - `_count`. Specifies how many records you want returned with the query.

The `/oaa/servlet` mapping assumes that you are using the default URL mapping that AssetCenter Web automatically defines for the Archway servlet. If you have defined another URL mapping, replace the servlet mapping with the appropriate mapping name.

The following are sample URL SQL queries:

- *host name/oaa/servlet/archway?sc.query&_table= probsummary&priority.code=1*

  This sends a query request to ServiceCenter for all records in the `probsummary` table with a priority code of 1.

- *host name/oaa/servlet/archway?ac.query&_table=amAsset&_return= Brand;mPrice;Model&_count=2*

  This sends a query request to AssetCenter for the first two records in the `amProduct` table. Only the **Brand**, **mPrice**, and **Model** fields are returned for each record.

The screen below shows the XML results of a query for products from AssetCenter.
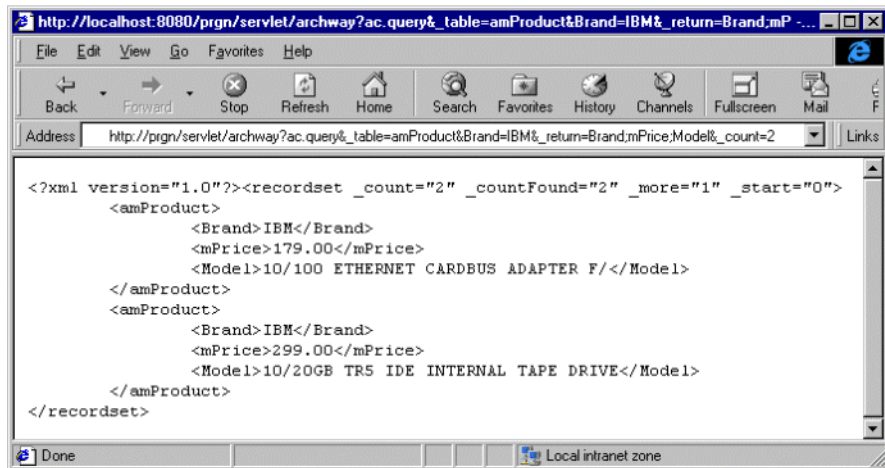
The Document Manager

Archway uses XML to exchange data and documents between clients and the supported back-end systems. Fundamentally, the XML data returned by Archway is obtained by executing queries against one or more systems. The queries can be executed by a direct URL request or indirectly within an ECMAScript function.

Simple queries are only capable of returning record sets of data. However, clients are more often interested in exchanging documents. A Document is a logical entity built up of several pieces of data that can come from various physical database sources.

The Document Manager uses schemas to determine which XML elements to use and what data should be contained in the elements. The data used by the Document Manager depends on the back-end system being used.

2 Customizing the Peregrine Portal

CHAPTER

Peregrine OAA provides a number of ways to customize the interface of an application built on the platform. You can make a quick change, such as replacing the logo with your company logo, or a more complex change such as rewriting the code that defines layer placement or frameset size.

This chapter includes advanced procedures for changing the Peregrine OAA Platform interface. To use this information effectively, you should have knowledge of XML and the CSS2 specifications established by the W3C as outlined at www.w3.org.

Topics in this chapter include:

- Deploying the Classic theme variations
- Changing the default theme
- Changing the header graphic for all themes
- Creating a custom theme
- Layers properties
- Changing framesets
- Creating script extensions
- Translating AssetCenter Web

Deploying the Classic theme variations

The Classic theme is the default theme that applications built on Peregrine OAA use. It has a gray and teal design and is the theme shown in all the screen shots in the guide. This is the theme you will use to create a customized theme for your enterprise.

There are four variations of the Classic theme:

Theme	Description
Accessible	Makes the screen available to users who need high contrast colors or better accessibility support. It provides 508 compliance.
Baja	Adds southwestern green and beige hues to the Classic design.
Quicksilver	Adds silver and blue hues to the Classic design.
Sierra	Adds teal hues to the Classic design.

These themes, as well as a number of other optional themes, are deployed with the application installation. Once you create your customized theme, Peregrine Systems recommends that you delete all other themes to prevent users from selecting one of them and overriding your custom theme. If you decide later that you want to manually deploy a theme that has been deleted, or if you did not deploy all themes during the installation, use the following procedure to deploy the themes. The additional themes are zip files located in the `C:\Program Files\Peregrine\oaa\packages` directory. You can identify the theme names from these zip file names.

To deploy an alternate Classic design:

1 Open a command prompt window, and change directories to your `oaa\packages` directory. The default path is:
`C:\Program Files\Peregrine\oaa\packages`

2 Type:

```
java -jar OAADeploy.jar <name of the theme>
```

Note: List each theme you want to deploy, separated by a space; for example, `java -jar OAADeploy.jar bluestheme hightechtheme bajatheme`.

3   Press ENTER.

4   Stop and restart your application server.

The themes you deployed appear as options the next time you log in to AssetCenter Web.

Changing the default theme

You can change the default theme that all users see when they log in to AssetCenter Web. Out-of-box the default theme is classic.

To change the default theme:

1   Open your Web browser and log in to the Admin module (`localhost/oaa/admin.jsp`) as the system administrator.

2   Click **Settings** > **Themes.** Change the following parameters:

- In the **Default skin/Theme** field, change the parameter to the name of the theme you want to use (for example, *Baja*).
- In the **Default style sheet** field, change the parameter to the appropriate name for the CSS file (for example, `baja.css`).
- In the **Default XSL templates** field, change the parameter to the name of the theme you want to use (for example, *Baja*).

3   Scroll to the bottom of the page, and then click **Save**.

4   Click the **Control Panel** link on the Admin Settings menu.

5   When the Control Panel opens, click **Reset Peregrine Portal**.

6   Refresh your browser to see the new default theme.

Changing the header graphic for all themes

You can add your corporate logo to all themes in the Peregrine OAA Platform from the Administration Settings page.

Warning: The administration setting discussed below overrides the image used by all themes. If you change this setting then you will see the same logo in all themes. If you want to use a different corporate logo for each theme, see Creating a custom theme on page 32.

To change the header graphic for all themes:

1 Create a custom header graphic.

Note: To fit within the default header frame, your customized header logo must be 514 pixels wide and 59 pixels high. If you want to change the header frame size, see Changing framesets on page 38.
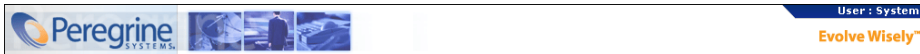
2 Save your custom header graphic to the following location:

```
C:\Program Files\Peregrine\Common\Tomcat4\\webapps\oaa\images\
skins\classic
```

Note: The Classic theme is the default theme.

3 Log in to the AssetCenter Web administration page (`admin.jsp`).

4 Click **Settings > Themes**.

5 In the **Default Peregrine Portal logo** field, type the name of your custom header logo.

Type your new image name.

6 Scroll to the bottom of the page, and then click **Save**.

7 Click the **Control Panel** link on the Admin Settings menu.

8 When the Control Panel opens, click **Reset Peregrine Portal**.

9 Refresh the browser to view your changes.

Creating a custom theme

You can create custom themes by copying and modifying the classic theme provided with AssetCenter Web.

To create a custom theme:

1 Copy classic theme images, style sheets, and XSL templates. These files are located at:

File type	Location
Images	`<application server>\oaa\images\skins\classic`
Style sheets	`<application server>\oaa\css\classic`
XSL templates	`<application server>\oaa\WEB-INF\templates\ classic`

2 Paste and then rename the folders for the classic theme to a new name. For example:

Images	`<application server>\oaa\images\skins\myTheme`
Style sheets	`<application server>\oaa\css\myTheme`
XSL templates	`<application server>\oaa\WEB-INF\templates\ myTheme`

3 Open and edit each image that you want to change in your new theme. Use the following image conventions.

- Image file names must remain the same. AssetCenter Web uses these image names to display theme elements.
- Image height and width should remain the same unless you are also changing the size of the framesets to accommodate new image sizes.

4 Open and edit the `classic.css` file in your new theme.

The following table lists some of the more commonly modified styles.

Style Name	Style Description
.ActionButton	The style used on buttons throughout the Portal.
.ActiveMenuLink	Used when the mouse hovers over a menu link.
.ActiveModuleMenu	Designates the currently-selected page within the navigational subset.
.CurrentModuleMenu	Designates the currently-selected navigational subset.
.FormTitle	Used for the title of forms. Normally used to title DocExplorer window content.
.ListboxEvenRow	A bolded version of TableEvenRow.
.ListboxHeading	A bolded version of Table Heading.
.ListboxOddRow	A bolded version of TableOddRow.
.MenuLink	Used within all module menus.
.ModuleMenu	Used for the left-hand navigational menu.
.ModuleMenuTitle	Designates the navigational subsets title.
.PageTitle	Used on the page title located directly below the logo and tabs.
.TableEvenRow	Used within the table heading with alternating background colors for ease of reading. Has a background color of white.
.TableHeading	Used for application headings for both search and results functions.
.TableOddRow	Used within the table heading with alternating background colors for ease of reading. Has a background color of light gray.
a.ListBoxEvenRow	Designates the style with a link attribute.
a.ListBoxOddRow	Designates the style with a link attribute.
a.TableEvenRow	Designates the style with a link attribute.
a.TableOddRow	Designates the style with a link attribute.

Tip: Modify the style sheets after you complete your overall theme design. Use your image editor's color picker to ensure that the your style sheet colors match your image colors.

Note: You can see a detailed style sheet key in the themes Administration section of the Portal. To access the style sheet key, locate the Default style

sheet field on the Themes tab of the Admin Settings page and click the
Peregrine Portal Stylesheet Key link.

Style Names	**Styles Used Throughout The Peregrine Portal**		
		How styles applied to HTML elements look.	
	<DIV><P> **<SPAN><TD> <TR>** with no Text	**<DIV><P>** **<SPAN><TD> <TR>** with unstyled text	**<FONT>** styled by itself
ActionBar		Text Sample	Text Sample
ActionButton		**Text Sample**	**Text Sample**
ActionSeparator		Text Sample	Text Sample
ActiveHeaderLink			
ActiveHeaderMenu		Text Sample	Text Sample
ActiveMenuLink		**Text Sample**	**Text Sample**
ActiveModuleMenu		Text Sample	Text Sample
ActiveTableRow		Text Sample	Text Sample
Body		Text Sample	Text Sample
BodyAlt		Text Sample	Text Sample
BodyFRSep		Text Sample	Text Sample
BodyFRSepAlt		Text Sample	Text Sample
BodyHead		Text Sample	Text Sample
BodyHeadAlt		Text Sample	Text Sample
CurrentModuleMenu		**Text Sample**	**Text Sample**
DebgTable		Text Sample	Text Sample
EntryTableFields		Text Sample	Text Sample
EntryTableHeading		**Text Sample**	**Text Sample**
EntryTableInstructions		Text Sample	Text Sample
EntryTableLabels		Text Sample	Text Sample
FieldLabel		**Text Sample**	**Text Sample**
FieldsHeading		Text Sample	Text Sample
FieldsTable		Text Sample	Text Sample
FieldTablePadding		Text Sample	Text Sample
FieldTableSeparator		Text Sample	Text Sample

5 Save your theme style sheet with the same name as your new theme. For
example, `<application server>\oaa\css\myTheme\myTheme.css`.

6 Open and edit the `layers_<xx>.jsp` file to change any layer descriptions.

To change layers for Internet Explorer, open `layers_ie.jsp`. To change
layers for Netscape open `layers_gecko.jsp` extension.

For more information, see Layers properties on page 36.

7 Open and edit any XSL style sheets you want to change.

Warning: Do not change these files unless you have knowledge of HTML and XSL transformations.

The XSL style sheets determine how AssetCenter Web displays form components in the main portal frame.

The following table lists the XSL style sheets you can change.

To change	edit this XSL stylesheet
Attachment picker	attachments.xsl
HTML form generation	basic-form.xsl
Action (button) properties	button.xsl
Template components	components.xsl
Debugging message properties	copy_nodes.xsl
Date-time picker properties	datetime.xsl
Text edit field properties	edit_fields.xsl
Entry table form component (see administration page for examples)	entrytable.xsl
Field section properties	fieldsection.xsl
Field table properties	fieldtable.xsl
HTML page generation	form.xsl
Frameset properties	frames.xsl
Images properties	image_fields.xsl
Label properties	labels.xsl
Link properties	link.xsl
Building of DocExplorer lists	list-builder.xsl
Lookup field properties	lookup_fields.xsl
Money text field properties	money_fields.xsl
Portal properties	portal.xsl
Radio checkbox properties	radio_checkbox_fields.xsl
Read-only text field properties	readonly_fields.xsl
Select text field properties	select_fields.xsl
Spinner properties	spinner_fields.xsl
SVG image properties	svg_cad.xsl
Table properties	table.xsl
Navigation tab properties	tabs.xsl

8 Stop and restart your application server.

You can view your new theme by selecting it from the *Change theme* page, available from the Peregrine Portal Home page.
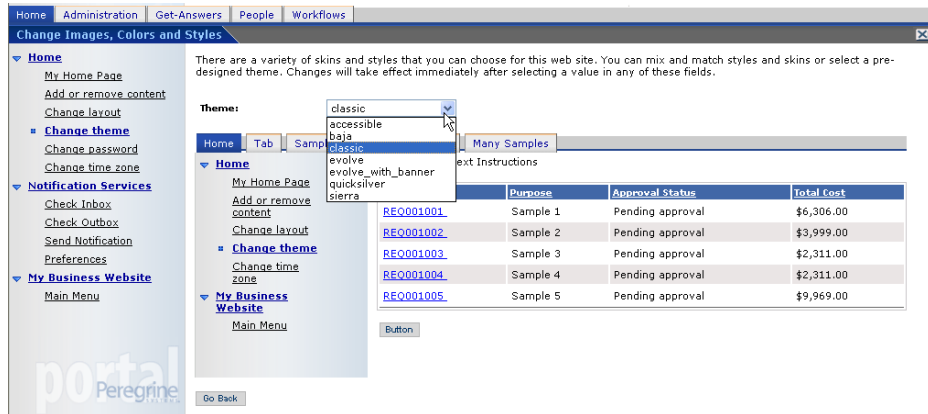
Layers properties

The following sections describe the `layers_ie.jsp` and `layers_gecko.jsp` files. Each layer is defined by a separate `<div>` tag entry and includes an `id` attribute that names the layer. You can change layer properties as needed, but the following layers are required and should not be removed.

logo

```
<div id="logo" style="position:absolute; left: 0px; top: 0px;
width: 100%; height: 40px; z-index: 3;">
<img name="logo" border="0" src="<%= logo %>" alt="logo"></div>
```

time

```
<div id="time" style="position:absolute; right: 4px; top: 84px;
width: 100%; z-index: 13;" onmouseover="_pauseAlert()"
onmouseout="_startAlert()"  class="userBarText">
</div>
```

toolbar

```
<div id="toolbar" style="position:absolute; width: 50px; top:
59px; right: 0px; z-index: 12;"></div>
```

user

```
<div id="user" style="position:absolute; top: -4px; right: 0px;
z-index: 14;">
<table width="100%" border="0" cellpadding="0" cellspacing="0"
align="right">
<tr>
<td width="50%"> </td>
<td nowrap width="3" align="right" valign="top">
<img src="<%= Archway.getSkinImagePath("backgrounds/rt_l.gif",
user ) %>">
</td>
<td nowrap align="right" valign="top" width="100%"
background="<%=
Archway.getSkinImagePath("backgrounds/rt_tile.gif", user ) %>">
<img src="<%= Archway.getSkinImagePath("backgrounds/rt_tile.gif",
user ) %>">
</td>
<td nowrap><font class="userBarText" size="1" face="Arial,
Helvetica, sans-serif"><%=userTitle%></font>  </td>
</tr>
</table>
</div>
```

tabs

```
<div id="tabs" style="position:absolute; left: 0px; top: 60px;
width: 100%; z-index: 11;" >
</div>
```

form titles

```
<div id="formTitles" style="position:absolute; left: 10px; top:
81px; width: 200px; z-index: 16;"> 
</div>
```

Changing framesets

Important: You must have advanced knowledge of HTML, JSP, and framesets to modify these files. Keep all of the frames and do not change the names of any of the frames. Doing so will result in JavaScript errors.

There are two framesets to be modified for each browser. These files are in C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\images \skins\<your theme>.

The frames_xx.jsp files are for the pages that you access when logging in as an end-user (login.jsp). The admin_frames_xx.jsp files contain the configuration for the Admin module (accessed when you log in using admin.jsp).

To change framesets:

1 Stop your application server.

2 Open the browser-specific frameset file frames_<xx>.jsp in a text editor (where <xx> is ie for Internet Explorer and gecko for Netscape).

3 Modify the frameset properties.

4 Save the file.

5 Restart your application server.

You can now test your changes in your Web browser.

The following sections show the complete _ie.jsp files as examples of the frameset files.

frames_ie.jsp

```
<%@ include file="../../../jspheader_2.jsp" %>
<%@ include file="../../../message_special.jsp" %>

<frameset onload="setTopFrames()" onunload="closeChildWindows()"
border="0" framespacing="0" frameborder="NO" cols="*"
rows="102,*">
   <frame scrolling="NO" marginwidth="0" marginheight="0"
src="oaa_header.jsp" name="getit_main_head">
      <frameset cols="185,10,*" rows="*" frameborder="no"
border="0" framespacing="0">
         <frame scrolling="AUTO" marginwidth="0" marginheight="0"
src="apphead.jsp" name="getit_header">
         <frame name="framesep" scrolling="no" marginheight="0"
marginwidth="0" src="framesep.jsp">
            <frameset rows="*,0">
               <frame scrolling="AUTO" marginwidth="6"
marginheight="6" src="e_login_main_start.jsp?<%=
user.getADW(msg,"Params" ) %>" name="getit_main">
               <frame noresize scrolling="NO" marginwidth="0"
marginheight="0" src="backchannel.htm" name="backchannel">
            </frameset>
      </frameset>
</frameset>
```

admin_frames_ie.jsp

```
<%@ include file="../../../jspheader_2.jsp" %>
<%@ include file="../../../message_special.jsp" %>

<frameset onload="setTopFrames()" onunload="closeChildWindows()"
border="0" framespacing="0" frameborder="NO" cols="*"
rows="102,*">
   <frame scrolling="NO" marginwidth="0" marginheight="0"
src="oaa_header.jsp" name="getit_main_head">
      <frameset cols="185,10,*" rows="*" frameborder="no"
border="0" framespacing="0">
         <frame scrolling="AUTO" marginwidth="0" marginheight="0"
src="apphead.jsp" name="getit_header">
         <frame name="framesep" scrolling="no" marginheight="0"
marginwidth="0" src="framesep.jsp">
            <frameset rows="*,0">
               <frame scrolling="AUTO" marginwidth="6"
marginheight="6" src="e_adminlogin_login_start.jsp?<%=
user.getADW(msg, "Params") %>" name="getit_main">
               <frame noresize scrolling="NO" marginwidth="0"
marginheight="0" src="backchannel.htm" name="backchannel">
            </frameset>
      </frameset>
</frameset>
```

Creating script extensions

By creating ECMA script extensions, you can modify the actions of the
out-of-box script without changing the original script. Extension scripts are
added to the `jscriptextensions` directory under `WEB-INF/apps/<component>`,
where `<component>` is the name of an application module (for example, `common`,
`portal`, and so on).

When adding extension scripts, you must:

- Preserve the hierarchy (path) of the out-of-box script under the
  `jscriptextensions` directory.
- Use only one extension per OAA `.js` file.

The extension file can declare new functions as well as functions that override functions of the same name in the out-of-box file. The object created to represent the out-of-box file is configured as the *prototype* object of the extension object.

You can call original function implementations from a function that overrides the original by appending `proto` to the function reference. For example, to call the out-of-box implementation of `login.login()`, use `proto.login.login(msg);` from the `login()` method declared in the extension file.

The following example writes a message to the archway log when the user logs in.

```
function login( msg )
{
  env.log("*** YOUR MESSAGE GOES HERE*** ");
  return proto.login.login(msg);
}
```

Note: The relative path of the example extension file is
       `..\WEB-INF\apps\common\jscriptextensions\login.js`.

If the out-of-box method calls other functions that are also overridden in the extension file, use the following technique to ensure that the new implementations of the secondary functions are called:

`proto.login.login.apply(this, arguments);`

where `arguments` is an implicit variable that exists in the context of an ECMA Script function.

```
function login( msg )
{
  env.log("*** YOUR MESSAGE GOES HERE*** ");
  return proto.login.login.apply(this, arguments);
}
```

Translating AssetCenter Web

Out-of-box, all Peregrine OAA web applications are provided in English. You can download the AssetCenter Web language pack from the Peregrine Customer Support Web site 90 days after the English release. Language packs are available for the following languages:

- French
- Italian
- German

Note: Not all Peregrine OAA web applications offer language packs. Refer to the Peregrine support web site to determine the availability of language packs for your Peregrine OAA web applications.

If you have a language pack version of a Peregrine OAA web application, you will need to edit the existing string files for these applications and add any new strings that resulted from your tailoring efforts. For more information on the process, refer to Editing existing translation strings files on page 42.

If you do not have a language pack version of your Peregrine OAA web applications and you want to create a new translation, refer to the instructions in Adding new translation string files on page 43.

To configure the Peregrine OAA Platform to use your new translation, see To configure the Peregrine OAA Platform to use new string files: on page 44.

Editing existing translation strings files

You can make edits, additions, and deletions to string files using any text editor or standard translation software.

To edit an existing translation string file:

1   Open the translated string file in a text editor or translation program.

    You can find all the translation string files in your application server's
    installation directory:

    ```
    <application server>\oaa\WEB-INF\apps\<group of modules name>
    ```

Note: The string file will use the ISO-639 two letter abbreviation for the
      language in the file name.

1   Search for an existing string that you want to change.

    The string file uses the format illustrated below:

    ```
    String_label, "translated string"
    ```

    Where `String_label` is a unique name given to the string, and

    Where `translated string` is the actual value of the string to be translated.

    For example if you added a new button, you might look for:

    ```
    EMPLOOKUP_EMPLOYEELOOKUP_SEARCH_LABEL, "Search"
    ```

2   Change the "`translated string`" portion of the new string to the target
    language of your translation. For example, to change the string listed above
    to French, you might enter the following:

    ```
    EMPLOOKUP_EMPLOYEELOOKUP_SEARCH_LABEL, "Recherche"
    ```

3   Save the new string file.

    The new translation strings will be available as soon as you stop and restart
    the application server.

Adding new translation string files

You can add new string files to the Peregrine OAA Platform to provide
additional language support to your Get-It web applications. The translation
process can be accomplished using any text editor or standard translation
software.

Important: Peregrine does not support Get-It web applications translated into any languages other than those listed in the section .

To add an existing translation string file:

1 Open the English string file for your Studio project in a text editor or translation program.

You can find all the translation string files in your application server's installation directory:

```
<application server>\oaa\WEB-INF\strings
```

Note: The English string file will have the ISO-639 two letter abbreviation EN in the file name.

2 Copy the entire the English string file.

3 Create a new string file for the target language in which you want to add a translation.

Note: The string file must use the ISO two letter abbreviation for the language in the file name.

4 Paste the copied English string file into the new file.

5 Change the "translated string" portion of each string to the target language of your translation.

6 Save the new string file.

The new translation strings will be available as soon as you stop and restart the application server.

To configure the Peregrine OAA Platform to use new string files:

1 Log in as an administrator (the administrator login page is located at admin.jsp).

2 Click **Settings**.

3 Click the **Common** tab.

4 Enter the two letter ISO-639 language code for the languages you want to support in the **Locales** field. The first code entered will be the default language used. The other languages you define will be available in a drop-down list.

5 In the **Content type encoding** field, enter the character encoding to be used for the display language. The following table lists some of the common character encoding formats.

Note: Peregrine recommends using the default setting, UTF-8, as UTF-8 encoding supports all of the other character sets available.

Character Encoding	Character Set
UTF-8	Supports all of the other encoding available.
ISO-8859-1	U.S. and Western European character sets. This is the default character set used by Studio.
Shift_JIS	Japanese character set
ISO-8859-2	Polish and Czech character set

6 Click **Save** at the bottom of the Settings form to save your changes.

7 On the Console form, click **Reset Peregrine Portal** to implement your changes.

Users will now be able to select the display language for their session used when they login to the Peregrine OAA Platform.

CHAPTER

3 | Using the Peregrine Portal

The Peregrine OAA Platform includes a Navigation menu, an Activity menu, and buttons that enable you to customize your Portal and to end your session.

Your installed Web applications determine the contents of the Navigation menu. However, if you log in as an administrator, all Navigation menus include an Administration tab that provides access to the Admin module.

The graphics in this chapter use the Classic stylesheet and are examples of a generic interface. Also, the Admin module displays only those features that AssetCenter Web uses.

Topics in this chapter include:

- Logging in to the Peregrine Portal on page 47
- Using the Activity menu on page 49
- Personalizing the Peregrine OAA Platform on page 50

Logging in to the Peregrine Portal

There are two login screens that provide access to the Peregrine Portal:

Login screen	URL
User login	*http://<server>/oaa/login.jsp*
Administrator login	*http://<server>/oaa/admin.jsp*

Note: An alternative to this login method is the Integrated Windows Authentication. See the Security chapter of this guide.

Note: This chapter discusses the features available with a user login. For more information about the administrator login, see the chapter on AssetCenter Web Administration in this guide.

The following is an example of the user login interface. Enter your user name and password. In the Language pull-down list select the language that you want the Peregrine Portal to display. English is the default login language, but you can enable other languages in the Settings page of the Administration Control Panel. For more information about enabling login languages, see the section Choosing a Login Language.

The following graphic shows a Portal without any applications installed. The Navigation menu includes modules for your particular application. All applications have the Admin module.

Activity menu   Navigation menu   Personalize your Home page   End your session
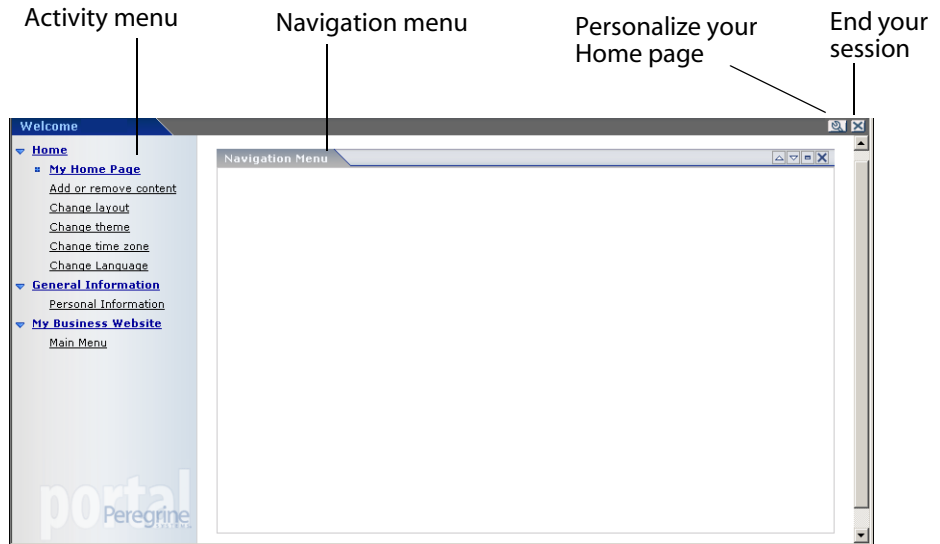
Using the Activity menu

The Activity menu provides access to a number of tasks as you navigate through your Web application. The menu remains visible as you change screens.

The default Activity menu includes the following choices:

Use this option	When you want to
My Home Page	Return to the Peregrine Portal Home page.
Add or remove content	Access the same page as the **Personalization** button, allowing you to customize your Home page.
Change layout	Change the location of a component or remove it from the Peregrine Portal.
Change theme	Select from several options. Changes take effect immediately after selecting a value in any of these fields. **Note:** Select the **accessible** theme to access the alternate text-based interface.
Change time zone	Select the time zone.

Personalizing the Peregrine OAA Platform

By default, the Navigation menu is displayed on the Peregrine Portal. You can personalize the Peregrine Portal to add AssetCenter Web utilities as well as personal tools such as a calendar, calculator, or the date and time. You can also change the layout of these components or minimize a component to hide the component details.

See the chapter on Using the Personalization Interface in this guide for more information on personalization.

Adding components

The following components are available.

Personal Utilities

This component	Provides
Calculator	A tool using standard arithmetic functions.
Calendar	A monthly calendar.
Theme Selector	A drop-down list to change themes.
Date and Time	A date and time display for the local time zone.

Peregrine Portal Web application components

This component	Provides
Navigation Menu	Quick links to the various modules that make up this application.
Document List	A display of a document search, list, or detail screen. Configure the component by choosing the document type you want to expose and the type of screen desired.
My Menu	A menu of links that can be configured dynamically. Links can point to arbitrary web sites, other menus, or document explorer screens.

Note: The Calendar and Calculator require Microsoft Internet Explorer 6.0+ or Netscape 6.1+.

Administration components

Only users with Admin capability have access to the Admin components.

This component	Provides
Connection Status	A list of the adapters currently registered in this server and their connection status.
Control Panel	A button to reset the server and all its connections.
Page Hits / Minute	A list of the total number of pages accessed per minute.
Adapter Transactions / Minute	A list of the number of transactions performed against adapters.
Active User Sessions	A list containing the number of active user sessions.

To add Peregrine Portal components:

1 Click the **Personalize** (wrench) icon.

Note: You can also select the **Add or remove content** link from the Activity menu.

The **Customize My Home Page** opens with a list of the components available for customizing.
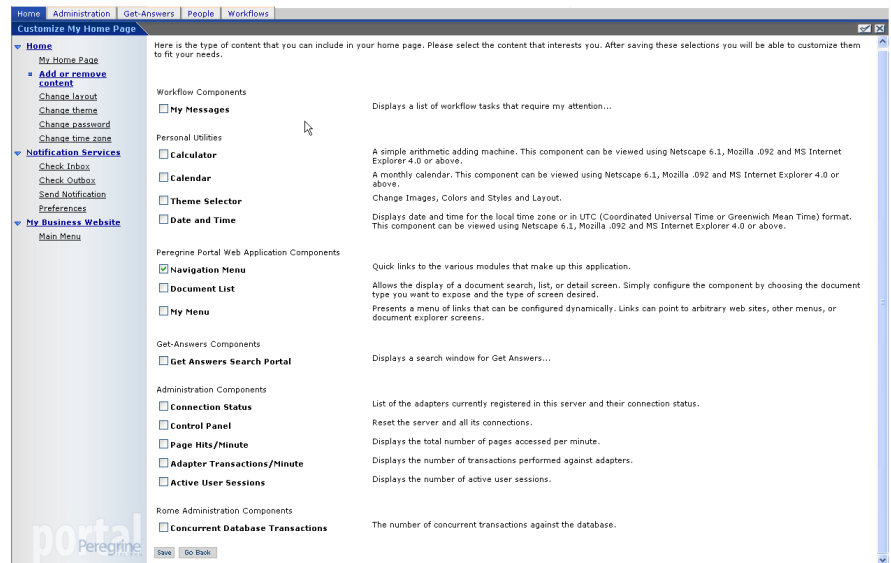
2   Select the components you want to either add or remove from your Peregrine Portal.

3   When you complete your selections, scroll to the bottom of the page, and then click **Save**. To return to the Peregrine Portal without making any changes, click **Go Back**.

When you return to the Peregrine Portal, the new components appear in the Peregrine Portal. The following example shows the .

Changing the layout

The following sections contain procedures for changing the location of the components or removing them from the Peregrine Portal. Your Web browser determines the procedure you use.

Microsoft Internet Explorer

If you are using Microsoft Internet Explorer as your Web browser, use the buttons in the upper right corner of each component to move the component up or down, remove the component, or hide/show the component detail.

Hide/Show component
detail

Move
component up

Remove
component

Move component down
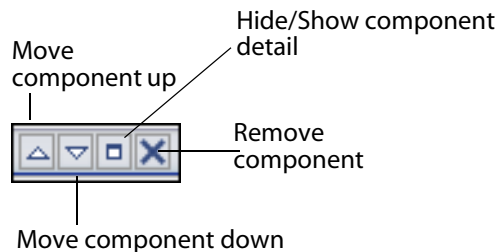
The following screen shows the Calculator hidden or minimized.

Netscape Navigator

If you are using Netscape Navigator as your Web browser, use the following procedure to change the status of the components on the Peregrine Portal. You can move a component up or down, or remove the component.

1 From the Activity menu, select **Change layout**.

A **Change Layout** page opens where you select the components you want to change.
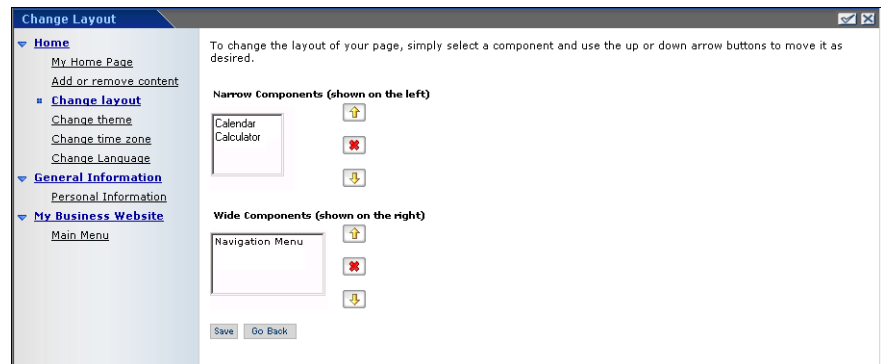
Components can be Narrow (for example, Calendar or Calculator) and are on the left side of the Peregrine Portal. Other components (for example, Navigation Menu) are Wide and are on the right side of the Peregrine Portal.

2 Select the component you want to modify, and then click the appropriate button to activate the change.

  ■ Up arrow moves the component up.

  ■ Down arrow moves the component down.

  ■ X removes the component from the Peregrine Portal.

3 Click **Save**.

Changing themes

You can choose from a number of themes to change the look of your Web pages. The of-the-box installation provides several themes from which you can choose. If you want to deploy additional themes, refer to Customizing the Peregrine Portal.

To change the theme:

1 From the Activity menu on the Portal Home page, select **Change theme**.

2 Choose a theme from the drop-down list.

As soon as you make your selection, the page updates to reflect your selection. The following example shows the Sierra theme.

This new configuration remains through subsequent work sessions for the user until changed by the user.

Displaying form information

You can view information about the form you are using. Set this parameter from the Logging tab on the Settings page of the Admin module. See the AssetCenter Web Administration chapter in this guide for more information.

When the **Show form info** parameter is set to Yes, a **Display Form Info** button appears on the upper right corner of forms.

The Display Form Info button shows information about the form you are using.
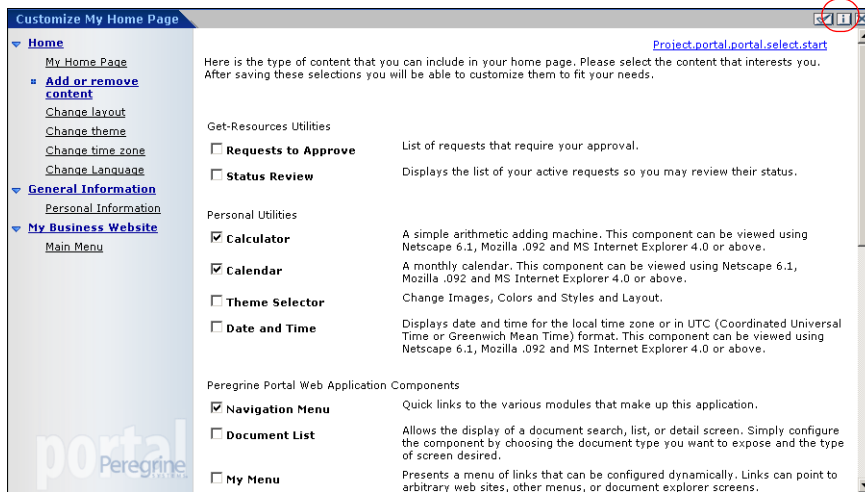
4 Using the Personalization Interface

CHAPTER

Personalization is available to both administrators and end users in Peregrine Systems Web applications built using Document Explorers (Doc Explorers). Authorized users can change the appearance and functionality of certain Web applications directly from the application interface.

With a Personalization interface, users can add or remove fields, rearrange how fields display, or add, change, or delete records from a back-end database.

This section includes:

Personalization overview

Personalization allows end users a means to create and customize searches of AssetCenter Web data. From the end-user perspective, personalization is a collection of standard forms that allow users to change part of the interface to suit their needs. The administrator determines which forms and features of personalization each user has by setting global personalization rights and by granting individual users capability words to do additional personalization.

From the administrator's perspective, personalization is a customization option that allows users to change the AssetCenter Web interface. Personalization

enables users to add or remove fields, change the layout of a form, and change interface elements such as headers and buttons in real time using the browser interface.

Forms and functions

Personalization is based on a collection of forms collectively referred to as a DocExplorer. DocExplorer forms provide the following functionality:

- A search form for defining search criteria.
- A list form for displaying search results.
- A details form for displaying detailed information about search results.

If you grant end-users administrative rights, they can also use Personalization for the following actions:

- Add a new record to the database from a creation form.
- Update existing records in the database from the detail form.
- Delete existing records from the database from the detail form.

Personalization interface

You can personalize any Web application interface that displays a wrench icon in the top right of the interface frame. The wrench icon only appears on forms that allow personalization. The Personalization form determines what options appear in the Personalization pop-up window.

Note: The graphics used in this section are intended as examples only of the personalization options available on different forms and may not match exactly the options available for your application.

When you click on the Personalization icon, a pop-up window opens showing the current settings for the form you are viewing.

The Available Fields column contains all the fields that can be added to a form.

The Current Configuration column contains the fields that currently visible on the form.

Administrators determine what available data fields display on each form. The Personalization form determines what options display in the Personalization pop-up window.

The form Personalization pop-up windows have the following format.

Field	Description
Available Fields	Shows all the document fields and sub-document collections that can be added to the current form. Peregrine OAA generates the list of available fields by dynamically reading the schema that the form uses. Any items listed between dashes are form components you can use to organize and arrange how document fields are displayed in the form.
Current Configuration	Shows all the document fields, subdocument collections, and displays components that are on the current form.
Form Options 　　Title 　　Instructions	Defines the form name and specific instructions to follow when using the form. The $$IDS provides a lookup function that assoociates string variables with strings in a particular language. The file and instruction text is found in a language-specific file under the WEB-INF\apps directory.
Explorer Options 　　Create 　　Skip search 　　Single Detail 　　Summary	Defines how Peregrine OAA displays results. Users only see the Options section if they have administrative Personalization rights.
Restrict operations to the following roles 　　Document Creation 　　Document Deletion 　　Document Update	Determines whether users can update, create, or delete records from the back-end database system. Users only see the Restrict section if they have administrative Personalization rights.
Revert to Default	Removes all Personalization entered by the end user and returns the form to the default state as determined by the AssetCenter Web administrator or the form's schema.
Save	Saves and applies your Personalization changes to the current form.

Adding and removing personalizations

You personalize the AssetCenter Web pages by adding, moving, and removing fields. Select the page you want to personalize, then select the fields you want to display on the screen.

Choose a row in the Available Fields list and use the appropriate icon below to add or insert an element:

Icon	Description
	The Plus (+) icon adds a component to your current configuration.
	The Insert icon adds a component in the specified place on the screen.

Choose a row in the Current Configuration list and use the appropriate icon below to customize the form layout:

Icon	Description
	The Personalize (wrench) icon allows you to edit any attributes available for the field. The personalization options available depend on the type of element selected.
	See the following sections for more information: Configuring fields on page 62; Configuring subdocuments on page 63; and Configuring collections on page 65.
	The Move (vertical arrows) icons move the component either up or down on the page. Moving components is always performed with these arrows.
	The Remove (X) icon removes the component from the page.
	Removing a component does not delete the component; it only does not display it.

The Available Fields list also includes positioning features that allow you to group date on the form layout:

Icon	Description
-- Left/Right Split --	Create verticle columns at the insertion point.
-- Top/Bottom Split --	Create horizontal splits between fields at the insertion point.
--Section Title --	Create an editable section title name for a group of fields.

Configuring fields

After you select a field in the Current Configuration list, click the wrench icon to configure the field's attributes:

See the section Configuring field attributes on page 72 for more information about specific tasks. Depending on your personalization rights, you can rename the field label (see page 72), change the field to read-only (see page 73), require users to enter a value (see page 73), and change the size and span of a field (see page 74).

Configuring subdocuments

 Select a subdocument in the Current Configuration list and click the wrench icon to configure the subdocument.

Note: The Current Configuration list displays both fields and subdocuments. Not all forms include a subdocument.

When you personalize subdocuments, you have different Explorer Options than with higher-level interfaces.

Subdocuments that can be personalized contain the same icons as higher-level interfaces, including the wrench icon. However, subdocuments offer slightly different options.

Field	Description
Available Fields	Shows all the document columns that can be added to the current form. Peregrine OAA generates the list of available columns by dynamically reading the schema that the form uses. Any items listed between dashes are form components you can use to organize and arrange how document columns are displayed in the form.
Current Configuration	Shows the document columns and displays components that are on the current form.

Form Options
    Title
    Instructions
	Defines the form name and specific instructions to follow when using the form.

Explorer Options
    Lookup
    Popup
    Readonly
    Clear
    Required
    Exclude Drilldown
    Lookup Label
    Popup height
    Popup width
	Defines how Peregrine OAA displays results. Users see the Explorer Options section only if they have administrative Personalization rights.

Save	Saves and applies your Personalization changes to the current form.

Note: The first field for a subdocument is always used by the lookup to display the returned value from the lookup activity. If you do not want the lookup icon and link to appear at all, unchecked Lookup, clear all check box options, and clear the Label field for this subdocument. The remaining fields specified on the subdocument personalization form appear as fields on the current form. If Readonly is checked for the subdocument, then the fields are displayed as read-only.

Configuring collections

After you select a collection in the Current Configuration list, click the wrench icon to configure the collection.

When you personalize collections, you have different Explorer Options than with higher-level interfaces.

Collections that can be personalized contain the same icons as higher-level interfaces, including the wrench icon. However, collections offer slightly different options.

Field	Description
Available Fields	Shows all the document columns that can be added to the current form. Peregrine OAA generates the list of available columns by dynamically reading the schema that the form uses. Any items listed between dashes are form components you can use to organize and arrange how document columns are displayed in the form.
Current Configuration	Shows the document columns and displays components that are on the current form.

Field	Description
Explorer Options Lookup Create Popup Remove Max. Row count More page row count Label Popup height Popup width	Defines how Peregrine OAA displays results. Users see the Explorer Options section only if they have administrative Personalization rights.
Save	Saves and applies your Personalization changes to the current form.

Note: A collection is handled as a many-to-many relationship if the first column of the collection is from another document. When personalizing a collection where you do not want a many-to-many relationship, make sure that its first column is not a subdocument reference; the first column in the collection must be a local attribute of the collected schema.

Supporting personalization

To support Personalization, you must have these components:

- An AssetCenter, ServiceCenter or Rome back-end database. Personalization requires you to store users' login rights and Personalization changes in one of these two databases.

- Adapter aliases defined for the following tabs on the AssetCenter Web Administration settings page:

  - Portal

  - PortalDB

  - Web Application

Activating Personalization

Personalization is an administration tool. Administrators can add and remove the fields they want to display in the interface and then turn off personalization to prevent end-users from adding or removing fields.

If end-users have personalization access, there is no way to prevent them from changing fields available through personalization.

You can grant users access to personalization features in one of two ways:

- Grant all users personalization rights by setting the end-user personalization administrative setting.
- Grant individual users personalization rights by adding a capability word to their user profile.

Granting global personalization rights

You can globally define end-user access to personalization by selecting one of three options from the End User Personalization options.

To grant all users personalization rights:

1   Login to the AssetCenter Web administration page.

2   Click **Administration** > **Settings**.

3 Click the **Common** tab and scroll down to the **End user personalization** parameter.

4 Select the level of personalization you want to grant all users from the **End user personalization** drop-down list.
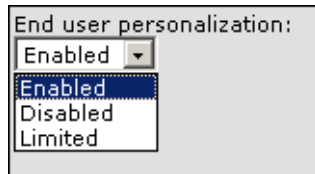
Personalization level	Description
Enabled	This setting grants all users the `getit.personalization.default` capability word, which enables end-users to add or remove any field listed in the schema used by a DocExplorer. However, only end-users who also have the `getit.personalization.admin` (or equivalent) capability word are able to use the advanced explorer options.
Disabled	This setting globally turns off all personalization except to users who the administrator has granted individual personalization rights by adding a capability word to their user profile in the AssetCenter Web back-end database. The personalization wrench icon is hidden from the AssetCenter Web interface, and any end-users who have individual personalization rights only see the fields that an administrator has configured to be visible.
Limited	This setting grants all users the `getit.personalization.limited` capability word, which enables end-users to add or remove only the fields that appear on a form by default or that administrator has made visible. Unless end-users have an individual capability word with greater rights, end-users can only add or remove the fields that an administrator has configured to be visible. This setting also prevents end-users from changing read-only fields to editable fields.

Tip: Peregrine Systems recommends that personalization be restricted to administrators in the production environment. That is, set the global Personalization setting to Disabled and then add the capability word **getit.personalization.admin** to the administrative user.

Granting individual personalization rights

You can grant individual users personalization rights by adding a capability word to the user profile stored in the AssetCenter Webback-end database. The following personalization capabilities words are available:

Capability word	Description
getit.personalization.limited	Users can only personalize features that have been exposed by a user with greater personalization rights.
getit.personalization.default	Users can change the layout and add or remove fields from the AssetCenter Web interface.
getit.personalization.admin	Users can do everything that the default capability word allows plus users can set personalization options and save personalization changes as the default layout. Any changes this user makes are global. The admin capability word also grants the following rights:
Document Creation	Users can specify the capability words required to create new records in the back-end database.
Document Update	Users can specify the capability words required to submit records to the back-end database.
Document Deletion	Users can specify the capability words required to delete records from the back-end database.
Save	Any personalization changes that the admin user saves determine what other users see. If the admin user adds a field, then the field becomes visible in the available fields list for other users. If the admin user removes a field, then the field is hidden from other users.

By default, users have no personalization capability word. To add a capability word, you must add it to the AssetCenter Web back-end database or set global personalization rights.

Note: The user that has the capability word **getit.portaladmin** creates the default portal page for all users that do not also have the **getit.portaladmin** capability word. There is one exception, however. The user "Admin", despite not having the **getit.portaladmin** capability word functions as if it does and can change the default portal settings as well.

Personalization tasks

Using DocExplorer, you can personalize any Web application interface that has a wrench icon in the top right of the Peregrine OAA frame. DocExplorers allow end users a means to create and customize searches of data. From the end-user perspective, a DocExplorer is a special activity that allows someone to personalize part of the interface. The user's profile determines the Personalization rights granted.

Adding fields to a form

With personalization rights, you can add fields to a form from the Available Fields list provided personalization is available for the form. You can then change the layout, if needed. Your personalization rights determine the lists of fields that you see.

You can add a field that is not currently available in the DocExplorer's schema by creating a schema extension. See the Document Schema Definitions chapter in this guide for more information on adding a new field.

Note: Data is not displayed in newly added DocExplorer fields. Users must close and resubmit the search or detail query before data appears in a new DocExplorer field.

To add fields to a form:

1   Do one of the following:

   ■   From the upper right corner of the active form, click the **Personalize** icon.

   ■   From the lookup page, click **Personalize this page**.

2   Select a field from the **Available Fields** list.

3   Click the **Plus** (+) icon.

   The field appears in the Current Configuration list.

4 Optionally, click the Insert icon to insert a component.

5 Click **Save**.

Tip: The browser warns that data must be present when adding fields in DocExplorer. Click Retry to resend the data to the browser. This is expected behavior of DocExplorer.

To arrange the field order:

1 Select a field from the Current Configuration list.

2 Click the up arrow or down arrow to change the field's position in the Current Configuration list.

3 Click **Save**.

To change the field layout:

1 From the **Available Fields** list, select **Left/Right Split**.

2 Click the **Plus** (+) icon.

To add a new section:

1 From the **Available Fields** list, select **Section Title**.

2 Click the **Plus** (+) icon.

Note: See Changing the label of a field on page 72 for information on editing the **Section Title** field.

3 From the **Current Configuration** column, arrange the order of the section with the **Up**, **Down**, and **Remove** icons.

Note: These icons either move or delete a field. Deleting a field removes the item from the form.

4 Click **Save** to keep your changes and return to the form.

Configuring field attributes

Each field in a personalization form has its own set of attributes that you can modify.

To configure field attributes:

1   Double click a field from the **Current Configuration** list to open an edit window.

2   Enter the new field attributes.

Note: Each field has its own set of field attributes. The following table only lists the more common field attributes:

Field	Description
Column span	The number of data cells in a column.
Label	The name to be used as the field label. This name appears next to the field in the AssetCenter Web interface.
Readonly	**Yes** prevents users from updating information displayed in the field.
Required	**Yes** requires the field to have a value before the form can be submitted.
Row span	The number of data cells in a row.
Size	The number measurement of a component in a cell.

3   Click **Save** to save your changes and return to the previous pages.

Cancel returns you to the previous page without saving your changes.

Changing the label of a field

To change the label of a field:

1   From the **Current Configuration** column, select the label you want to change.

2 Click the **Personalize** (wrench) icon.

The Personalization window opens.

3 Type the new name in the **Label** text box, then click **Save** to save your changes and return to the previous pages.

Cancel returns you to the previous page without saving your changes.

Changing a field to read-only

You can make a field read-only if you do not want users updating information in the displayed field.

To change a field to read-only:

1 From the **Current Configuration** column, select the field you want to be read-only.

2 Click the **Personalize** (wrench) icon.

The Personalization window opens.

3 Toggle the **Read Only** field to **Yes**.

4 Click **Save** to save your changes and return to the previous pages.

Cancel returns you to the previous page without saving your changes.

Making a field required

You can require users to enter a value in a field before they can submit a form.

To make a field required:

1 From the **Current Configuration** column, select the field you want to be required.

2   Click the **Personalize** (wrench) icon.

The Personalization window opens.

3   Toggle the **Required** field to **Yes**.

4   Click **Save** to save your changes and return to the previous pages.

Cancel returns you to the previous page without saving your changes.

Changing the size and span of a field

You can change the dimensions of the field by assigning values to the row span and size.

To change the size and span of a field:

1   From the **Current Configuration** column, select the field you want to change.

2   Click the **Personalize** (wrench) icon.

The Personalization window opens.

3   Type the values for the **Row Span** and **Size**.

4   Click **Save** to save your changes and return to the previous pages.

Cancel returns you to the previous page without saving your changes.

Removing fields from a form

To remove fields from a form:

1   Select a field from the **Current Configuration** list.

2   Click the **X** button to remove the field.

3   Click **Save**.

Moving personalizations to a production environment

You can easily export personalizations that you created in a development environment and import them to a production environment.

Moving the files is a two-step process:

Step 1 Export the personalization files from your development environment.

Step 2 Import the personalization files to your production environment.

Note: The import/export for personalization is only available to users having both `getit.admin` and `getit.personalization.admin` capability words.

To export personalization files:

1 Log in to the development application server.

2 Click **Administration** > **Import/Export** on the Administration tab.

3 Type the path to an existing folder on the server, including a file name, to make the file available to the production server.

4 Click **Export**.

5 Manually copy the file from your development server to your production application server.

To import personalization files:

1 Login to the production application server.

2 Click **Administration** > **Import/Export** on the Administration tab.

3 Change the path and file name to the path and file name of the file you want to import.

4 Click **Import**.

5

CHAPTER

Document Schema Definitions

This section describes document schema definitions and explains how they map data between AssetCenter Web and the back-end database. In addition, this document discusses how to use schema extensions to add new logical and physical mappings to existing schemas.

This document covers the following topics:

Understanding document schema definitions

A document schema definition (also called a schema) is an XML file that instructs the Archway Document Manager how to query back-end databases and generate XML documents containing the query response. Schemas are mapping tools that determine which XML tags used in dynamically created documents map to the table and field names in a given back-end database. These generated XML documents provide the data that AssetCenter Web displays and processes.

All schemas consist of two types of definitions:

Definition type	Description
Base definitions	The schema entries that provide a logical mapping between the XML tags generated in a document query to the AssetCenter Web interface are collectively referred to as the schema base definitions. The Archway Document Manager uses the base definitions to generate XML tags based on the elements listed in the schema. The Archway Document Manager converts the name value listed in an `<attribute>` element into an XML tag of the same name.
Derived definitions	The schema entries that provide a physical mapping between the XML tags generated in a document query to the table and field names in the back-end database are collectively referred to as the schema derived definitions. The Archway Document Manager queries the tables and field names listed in the schema and creates an XML document with the results of the query. The Archway Document Manager converts the `table` and `field` values listed in the `<document>` and `<attribute>` elements into a SQL query.

Note: The document schema definitions used by Peregrine Studio are not the same as the schemas being proposed and developed by the W3C.

The base and derived definitions each have their own list of legal elements and attributes. For more information on schema elements and attributes and how to use them, refer to Schema elements and attributes on page 114.

Sample schema

The following are two sample schemas that you can use for as templates for your schema extension logical and physical mappings.

Logical mappings

The file \schema\extensions\sample.xml would list the schema extension logical mappings. Logical mappings always use name="base". The document name determines the schema name. This schema is sample.xml

```xml
<?xml version="1.0"?>
<schema>

<!--============================================================
Schema extension for logical mappings
==============================================================-->
   <documents name="base">
      <document name="sample">
         <attribute name="Id" type="number">
         <attribute name="contact" type="string" label="Contact"
/>
      </document>
   </documents>
</schema>
```

Physical mappings

The file \schema\extensions\\sample.xml would list the schema extension physical mappings. Physical mapping lists the adapter name. Physical mapping uses the same attribute elements.

```xml
<?xml version="1.0"?>
<schema>

<!--============================================================
===
Schema extension for physical appings
==============================================================
=-->
   <documents name="">
      <document name="sample" table="">
         <attribute name="Id" field="" />
         <attribute name="contact" field= "" />
      <document/>
   </documents>

</schema>
```

How to use schemas

In most cases you will access a schema through personalization where the list of available fields for personalization is determined by the schema. For more information about how to use personalization, refer to the Using the Personalization Interface chapter of this guide. If you want to change the fields that are available through personalization, you can create a schema extension.

A schema extension is a separate file listing only the changes you make to an existing schema's logical or physical mappings. For example, you could create a schema extension to provide updated physical mappings when you upgrade your back-end database. Creating schema extensions is the preferred method of tailoring schemas as your changes are stored in separate files that can be easily carried over during an upgrade.

If you need change a schema outside of personalization, then you will need to purchase the AssetCenter Web Tailoring Kit.

Schema extensions

You can create schema extensions to add new *logical* and *physical* mappings to your existing schemas. Schema extensions allow you to save any additional mappings in separate files that preserve the original schema files shipped by Peregrine Systems. This separate file organization ensures that any upgrades will not overwrite your tailoring changes.

When to use schema extensions

Schema extensions generally provide the most benefit when you use them to extend existing DocExplorerschemas. Extending a schema allows you to do the following tailoring tasks without the need to rebuild a project in Peregrine Studio:

- Add new fields to the Available Fields list.
- Hide existing fields from the Available Fields list.
- Change the label that a field displays in the Available Fields list.

- Change the list of forms where a field displays.
- Change the physical mapping of a field.
- Change the type of data a field stores.
- Add subdocuments to the personalization Available Fields list.

For instructions how to perform these schema extension tasks, see Creating schema extensions on page 81.

There are some application tailoring tasks where you must use Peregrine Studio. These tasks include:

- Call custom scripts from a schema.
- Change the Document Field (schema name) that a form component uses.
- Change the document field to a customized field or column in a non-DocExplorer form.
- Change the schema used by a DocExplorer.
- Add a new schema to your project.

Creating schema extensions

You can create schema extensions outside of Peregrine Studio using any Text editor. The following procedures outline the steps required to create a schema extension.

To create schema extensions:

Step 1 Identify the schema that you want to extend. See Identifying the schema to extend on page 82.

Step 2 Locate the schema file on the AssetCenter Web server. See Locating the schema on the server on page 82.

Step 3 Create the schema extension target folders and copy XML files. See Creating the schema extension target folders and files on page 83.

Step 4 Edit the schema extension files to support the features you want. See Editing the schema extension files on page 84.

Identifying the schema to extend

You can identify the schema used by a particular form directly from the AssetCenter Web interface. Typically each form uses only one schema, but in some cases a form will use a subdocument that references another schema. The following procedures will help you determine what schema a particular form uses.

To identify the schema used by a particular form:

1   Enable Display form information from the **Administration** > **Settings** > **Logging** tab page.

    The Form information button displays in the banner bar of the AssetCenter Web interface.

2   Browse to the form that you want to tailor.

3   Click the Display form information button.

    The form information window opens.

Locating the schema on the server

After you have determined the name of the schema you want to extend, you can find it using your operating system's file search function. The following guidelines are provided to help narrow down your search:

■   All schemas files have a .XML extension
■   All schemas files are stored in the `\apps\<module>\schema` folder of your application server's deployment directory. For example:
    `C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\WEB-INF\apps\\schema`

Creating the schema extension target folders and files

Schema extensions require two separate files in subdirectories of the same directory where you found the source schema. For example:

```
C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\WEB-INF\
apps\\schema
```

Schema extension type	Description
Logical mappings.	This file contains the schema base definitions. These definitions determine the logical names and labels used for each field. You must create this file in a sub folder of `schema` called `extensions`, and it must have the same name as the schema that it extends. For example: `schema\extensions\`.
Physical mappings	This file contains the schema derived definitions. These definitions determine the back-end database tables and fields to which each logical name physically maps. You must create this file in a sub folder of `extensions` that matches the adapter name to your back-end database, and it must have the same name as the schema that it extends. For example: `schema\extensions\\`.

To create the schema extension target folders and files:

1 Copy the schema XML source file. For example, `Request.xml`.

2   Create two new folders as follows:

Create an `extensions` folder in the same directory where you found the source schema. For example:

```
C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\WEB-INF\
apps\\schema\extensions
```

Create an `<adapter name>` folder in the `extension` folder.

For `<adapter name>`, enter the abbreviation of the adapter used to connect to your back-end database such as . For example:

```
C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\WEB-INF\
apps\\schema\extensions\
```

3   Paste a copy of the source schema file in each of the two folders you created.

Editing the schema extension files

The edits that you need to do to the schema extension files depend upon what features you are trying to include. The following sections outline what edits you need to perform for each feature.

- Adding a new field to the Available Fields list on page 85.
- Hiding an existing field from the Available Fields list on page 87.
- Changing the label a field displays in the Available Fields list on page 89.
- Changing the list of forms where a field is available or visible on page 90.
- Changing the physical mapping of a field on page 92.
- Changing the type of form component a field uses on page 94.
- Adding subdocuments to the Available Fields list on page 95.

Adding a new field to the Available Fields list

You can add a field to any form that uses personalization. New fields display as options in the personalization Available Fields list.

To add a new field to Available Fields list:

1   Open the schema extension file in the `extension` folder.

This file is for your schema extension logical mappings.

2   Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3   In the `<document>` section that remains, add a logical mapping `<attribute>` element for each field you want to add to the list of Available Fields.

You must add each `<attribute>` element between the `<document>` tags:

```
<documents name="base">
   <document name="schema">
      <attribute name="Contact" type="string" />
   </document>
</documents>
```

a   Add the required `name` and `type` attributes to each `<attribute>` element.

b   Add any optional attributes you want to use for each `<attribute>` element.

Refer to <attribute> on page 121 for additional information on the `<attribute>` element.

4   Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

5 Save the logical mappings schema extension file.

6 Open the schema extension file in the `<adapter name>` folder.

This file is for your schema extension physical mappings.

7 Delete all the base definitions listed in the top half of the original schema.

The base definitions section starts with the first `<documents name="base" …>` element and includes all entries up to the closing `</documents>` element.

8 Find the element `<documents>` that has the `name` and `version` attribute values that match the adapter you want to use. For example, `<documents name="" version="4">`.

If you cannot find a matching `<documents>` element entry for your adapter, you must create one. See <documents> on page 114 for more information on the requirements of a `<documents>` physical mapping.

9 Verify that the `<document>` element beneath your chosen adapter lists the proper table and connection attributes required for your new fields.

If the attributes are not what your new fields require, you must edit the attributes. See <document> on page 117 for more information on the requirements of a `<document>` physical mapping.

Important: If the `<document>` element contains a ServiceCenter event mappings for either the insert or update attributes, then you must edit the listed ServiceCenter event mapping before your new field will add or update records correctly in ServiceCenter. See your ServiceCenter documentation for instructions.

10 Beneath the `<document>` element, add one physical mapping `<attribute>` element for each entry you added in the logical mapping.

You must add each `<attribute>` element between the `<document>` tags:

```
<documents name="" version="4.0">
   <document name="schema" table="table1">
      <attribute name="Contact" field="contact_name" />
   </document>
</documents>
```

a Add the required `name` and `field` attributes for each entry you defined in the logical mapping.

b Add any optional attributes you want to use for the physical mapping.

See <attribute> on page 121 for more information on optional attributes of the `<attribute>` element.

11 Delete any other physical mappings that you will not be updating in this schema extension file.

Tip: List only the new physical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

12 Save the physical mappings schema extension file.

Hiding an existing field from the Available Fields list

You can hide a field from the list of Available Fields in personalized forms. Hidden fields will not be available to any user regardless of user rights.

To hide an existing field from the Available Fields list:

1 Open the schema extension file in the `extension` folder.

This file is for your schema extension logical mappings.

2   Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3   Locate the logical mapping for the field you want to remove.

Use the `label` attribute to identify the proper field. For example, if the DocExplorer Available Field you want to remove is called `Contact`, search the `<attribute>` element that has the value `label="Contact"`.

4   Add one of the following values to make a field available or visible.

```
search="false"
list="false"
detail="false"
create="false"
```

For example, the following settings will make the `contact` field both available and visible in all DocExplorer forms.

```
<documents name="base">
   <document name="schema">
      <attribute name="contact" label="Contact" search="false"
       list="false" detail="false" create="false" />
   </document>
</documents>
```

These settings tell DocExplorer to hide the field on the search, list, detail, and create forms.

5   Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

6   Save the logical mappings schema extension file.

7   If you will not be making any changes to the physical mappings in this schema, you may delete the schema extension file in the `<adapter name>` folder.

You only need to edit this file if you will define new physical mappings for your DocExplorer fields.

Changing the label a field displays in the Available Fields list

You can change the label that appears in the Available Fields list of personalized forms. Typically, you will only need to add labels to new fields that you have added to a schema.

To change the label a field displays in the Available Fields list:

1   Open the schema extension file in the `extension` folder.

You will define the logical mappings in this file.

2   Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3   Locate the logical mapping for the field you want to change.

Use the `label` attribute to identify the proper field. For example, if the DocExplorer Available Field you want to change is called `Contact`, search the `<attribute>` element that has the value `label="Contact"`.

4   Change the label attribute to the new desired value.

```
<documents name="base">
   <document name="schema">
      <attribute name="contact" type="string"
       label="Representative" />
   </document>
</documents>
```

5   Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

6   Save the logical mappings schema extension file.

7   If you will not be making any changes to the physical mappings in this schema, you may delete the schema extension file in the `<adapter name>` folder.

You only need to edit this file if you will define new physical mappings for your DocExplorer fields.

Note: The child schema does not inherit changes when you modify the label in the parent schema. To change the label in the child schema for that field, modify the schema string file (for example, `schema_en.str` for English) and modify the label name for the child schema.

Changing the list of forms where a field is available or visible

You can determine the list of DocExplorer forms in which a field is available or visible. By default, a new field is available in all DocExplorer forms, but not visible.

To change the list of forms where a field is available or visible:

1   Open the schema extension file in the `extension` folder.

You will define the logical mappings in this file.

2   Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3   Locate the logical mapping for the field you want to remove.

Use the `label` attribute to identify the proper field. For example, if the DocExplorer Available Field you want to remove is called `Contact`, search the `<attribute>` element that has the value `label="Contact"`.

4   Add one of the following values to make a field available or visible.

To make this form	Available	Visible	Neither available or visible
search	search= search =true	search=true	search=false
list	list= list=true	list=true	list=false
detail	detail= detail=true	detail=true	detail=false
create	create= create=true	create=true	create=false

For example, the following settings will make the `contact` field both available and visible in all DocExplorer forms:

```
<documents name="base">
   <document name="schema">
      <attribute name="contact" type="string" label="Contact"
       search="true" list="true" detail="true" create="true" />
   </document>
</documents>
```

5   Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

6 Save the logical mappings schema extension file.

7 If you will not be making any changes to the physical mappings in this schema, you may delete the schema extension file in the *<adapter name>* folder.

You only need to edit this file if you will define new physical mappings for your DocExplorer fields.

Changing the physical mapping of a field

You can change the physical mapping that a field uses to point to another back-end database, table, or physical field.

To change the physical mapping of a field:

1 Open the schema extension file in the `extension` folder.

You will define the logical mappings in this file.

2 Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3 Locate the logical mapping for the field whose physical mapping you want to change.

Use the `label` attribute to identify the proper field. For example, if the DocExplorer Available Field you want to change is called `Contact`, search the `<attribute>` element that has the value `label="Contact"`.

4 Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

5 Save the logical mappings schema extension file.

6 Open the schema extension file in the `<adapter name>` folder.

This file is for your schema extension physical mappings.

7 Delete all the base definitions listed in the top half of the original schema.

The base definitions section starts with the first
`<documents name="base" …>` element and includes all entries up to the first
`</documents>` element.

8 Find the element `<documents>` that has the name and version attribute
values that match the adapter you want to use. For example,
`<documents name="" version="4">`.

If you cannot find a matching `<documents>` element entry for your adapter,
you must create one. See <documents> on page 114 for more information
on the requirements of a `<documents>` physical mapping.

9 Verify that the `<document>` element beneath your chosen adapter lists the
proper table and connection attributes required for your new fields.

If the attributes are not what your new fields require, you must edit the
attributes. See <document> on page 117 for more information on the
requirements of a `<document>` physical mapping.

10 In the `<document>` section you selected, change the physical mapping `<attribute>` element to match the new physical mapping you want.

The physical mapping `<attribute>` elements are between the `<document>` tags:

```
<documents name="" version="4.0">
   <document name="schema" table="table1">
      <attribute name="Contact" field="contact_name" />
   </document>
</documents>
```

a Change the `field` attribute to the new physical mapping.

b Add any optional attributes you want to use for the physical mapping.

Refer to <attribute> on page 121 for more information on optional attributes of the `<attribute>` element.

11 Delete any other physical mappings that you will not be updating in this schema extension file.

Tip: List only the new physical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

12 Save the physical mappings schema extension file.

Changing the type of form component a field uses

You can change the type of form component a field uses by changing the type attribute value in a schema extension. For a list of all possible types and the form components they use, see <attribute> on page 121.

To change the type of form component a field uses:

1 Open the schema extension file in the `extension` folder.

You will define the logical mappings in this file.

2 Delete all the derived definitions listed in the bottom half of the original schema.

The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3 Locate the logical mapping for the field you want to change.

Use the `label` attribute to identify the proper field. For example, if the DocExplorer Available Field you want to change is called `Contact`, search the `<attribute>` element that has the value `label="Contact"`.

4 Change the type attribute to the new desired value.

```
<documents name="base">
    <document name="schema">
        <attribute name="contact" type="string" label="Contact" />
    </document>
</documents>
```

5 Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

6 Save the logical mappings schema extension file.

7 If you will not be making any changes to the physical mappings in this schema, you may delete the schema extension file in the *<adapter name>* folder.

You only need to edit this file if you will define new physical mappings for your DocExplorer fields.

Adding subdocuments to the Available Fields list

You can add a subdocument to add a lookup form component that references information from another schema. Subdocuments have two different formats depending upon the results returned by the schema query. For more

information on the schema elements and formats used with subdocuments, see Subdocuments on page 130.

To add subdocuments to the Available Fields list:

1   Open the schema extension file in the `extension` folder.

    This file is for your schema extension logical mappings.

2   Delete all the derived definitions listed in the bottom half of the original schema.

    The derived definitions section starts after the first `</documents>` element and usually has a comment section describing what back-end databases and versions the derivations apply to.

3 In the `<document>` section that remains, add one of the following sets of elements for each subdocument you want to add to the list of Available Fields:

Element	Condition for use	Subdocument requirements
`<document>`	Use if the subdocument query always returns *one and only one* result for each requested element in the subdocument. For example, a contact should only have one name.	Required attributes: name Optional attributes: docname
`<collection>`	Use if the subdocument query can return *more than one* result for each requested element in the subdocument. For example, a contact can have multiple open in his name.	Required attributes: name Required elements: `<document>`

```
<documents name="base">
   <document name="schema">
      <attribute name="contact" type="string" label="Contact"
/>
      ...
      <document name="address" docname="external_schema" />
      ...
      <collection name="telephone_numbers">
         <document name="telephone_number" />
      </collection>
      ...
   </document>
</documents>
```

4 Delete any other logical mappings that you will not be updating in the physical mapping schema extension file.

Tip: List only the new logical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

5 Save the logical mappings schema extension file.

10 Beneath the `<document>` element, add one of the following sets of elements for each logical subdocument that you added:

Element	Condition for use	Subdocument requirements

`<document>`	Use if the subdocument query always returns *one and only one* result for each requested element in the subdocument. For example, a contact should only have one name.	Required attributes:
		    table
		    field
		    joinfield
		    joinvalue
		
		Optional attributes:
		    docname

`<collection>`	Use if the subdocument query can return *more than one* result for each requested element in the subdocument. For example, a contact can have multiple open in his name.	Required attributes:
		    name
		
		Required elements:
		    `<document>`

```
<documents name="" version="4.0">
   <document name="schema" table="table1">
      <attribute name="contact" field="contact_name"/>
      ...
      <document name="address" table="table2"
joinfield="addressee"
       joinvalue="id" />
      ...
      <collection name="telephone_numbers">
         <document name="telephone_number" table="table3"
          joinfield="contact" joinvalue="id" />
      </collection>
      ...
   </document>
</documents>
```

11 Delete any other physical mappings that you will not be updating in this schema extension file.

> **Tip:** List only the new physical mappings in your schema extension files. Schema extension entries that duplicate entries in the source schema may reduce your system performance.

12 Save the physical mappings schema extension file.

Schema subclasses

A schema subclass is similar to a schema extension because it changes the default behavior of a schema by adding or removing schema elements. Unlike a schema extension however, a schema subclass only changes the default schema's behavior when it is specifically called in the context of a particular form or portal component that uses the specific subclass. You can use a schema subclass to override the normal schema behavior in one particular instance while preserving the normal schema behavior in all other contexts.

The following process describes how to create a schema subclass:

Step 1 Create the necessary folders to store your schema subclass and script files. See Creating necessary folders for a schema subclass on page 101.

Step 2 Create a `package.xml` file to add your custom files to your AssetCenter Web installation. See Creating a package.xml file on page 101.

Step 3 Create a `publicSchemas.xml` file to make your schema subclass visible to Document List and My Menu portal components. See Creating a publicSchemas.xml file on page 102.

Step 4 Edit the schema subclass files to support the features you want. Typically a schema subclass calls a custom loadscript. See Editing the schema subclass files on page 102.

Step 5 Create the custom loadscript used by your schema subclass. See Editing the loadscript files on page 104.

Creating necessary folders for a schema subclass

All schema subclass customizations must be saved in a separate folder. At a minimum you will need to create three new folders:

- A folder to store all of your customizations
- A folder to save schema customizations
- A folder to save script customizations

To create necessary folders for a schema subclass:

1   Open Windows Explorer and browse to the AssetCenter Web WEB-INF/apps folder in your application server. For example:

    ```
    C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\WEB-INF\
    apps
    ```

2   Create a folder to store all your customizations. For example:

    ```
    \custom
    ```

3   Browse to the new folder you created in Step 2 and create two new folders:

    - \schema
    - \jscript

Creating a package.xml file

The package.xml file lists all the tailoring changes that you have made.

To create a package.xml file:

1   Open a text editor such as Notepad.

2   Enter the following text:

    ```xml
    <?xml version="1.0" encoding="UTF-8"?>
    <Package>
    </Package>
    ```

3   Save the file as `package.xml` in the custom folder you created. For example:
    `\custom\package.xml`

Creating a publicSchemas.xml file

Listing your customizations in a `publicSchemas.xml` file makes them available to the Document List and My Menu portal components.

To create a publicSchemas.xml file:

1   Open a text editor such as Notepad.

2   Enter the following text:

```
<schemas>
    <schema>
       <document name="<Schema_subclass>" label="<Label_name>"/>
    </schema>
</schemas>
```

For `<Schema_subclass>`, enter the name you want your new schema subclass to have. This name must be unique from any other schema name.

For `<label_name>`, enter the name you want the schema subclass to have when it is displayed in the AssetCenter Web personalization interface.

3   Save the file as `publicSchemas.xml` in the custom folder you created. For example: `\custom\publicSchemas.xml`

Editing the schema subclass files

All schema subclass files require you to create a new schema file in your custom schema folder. The following general procedures illustrate how to create a schema subclass file that calls a loadscript file. Most of the actual customization you will do is done in the loadscript file called by your schema subclass.

To create a schema subclass file:

1   Open a text editor such as Notepad.

2   Create a new schema subclass of an existing schema file that has the fields you want to use. For example, to create a list of tickets filtered by the currently logged in contact, enter the following:

```
<?xml version="1.0"?>
<schema>

<documents name="base">
  <document name="tickets_by_contact"  label="Tickets by
contact"
    extends="Problem"
loadscript="tickets_by_contact.loadscript">
  </document>
</documents>
</schema>
```

The <document> extends attribute lists the original schema name for which you are creating a subclass. Your schema subclass you use all the properties of this existing schema except any entries that are listed in the schema subclass file.

The <document> loadscript attribute lists the script name you want to run with this schema subclass. Typically, a schema subclass runs a different loadscript than the original schema.

3   Save the schema subclass as an XML document in your custom schema subfolder. For example:

\custom\schema\tickets_by_contact.xml

Important: The schema subclass file name must match the value listed in the <document> name attribute.

4   Create a custom loadscript for your schema subclass.

Editing the loadscript files

The loadscript edits that you need to make files depend upon what features you are trying to include. The following sections outline what edits you need to make for each feature:

Filtering a list of documents in a portal component

You can create an automatically filtered list of documents in the Document List portal component by creating a schema subclass that defines a set filter criteria. For example, you can have a Document List that only displays tickets where the current user is listed as a contact. The Document List will display the filtered list every time you access the saved search in the portal component.

To filter list of documents in a portal component:

1 Open a text editor such a NotePad.

2   Create a new loadscript. For example, to get a filtered list of tickets by the currently logged in contact, enter the following load script:

```
import docExplorer;
import personalize;

function loadscript(msg)
{
  var explorer = personalize._getExplorer(
msg.get(DOCEXPLORER_CONTEXT), msg.get(DOCEXPLORER_INSTANCE) );
  var strAction = msg.get( DocExplorer.ACTION );

  // Example 1: Adding record list filtering criteria
  if ( strAction == PERSONALIZE_LIST )
  {
    msg.add( "tickets_by_contact/ContactName",
user.get("_name" ) );
  }

  // Call default the onload script
  var script=msg.get(DocExplorer.LOADSCRIPT);
  if ( script != "" )
    msg = env.execute(script, msg);

  return msg;
}
```

3   Save the loadscript as an JS file in your custom scripts subfolder. For example: `\custom\jscripts\tickets_by_contact.js`

Important: The load script file name must match the value listed in the `<document>` loadscript attribute of your schema subclass.

4   Stop and restart your application server to pick up your schema changes.

5   Login to AssetCenter Web and add a new Document List search on your portal page using the schema subclass you created.

Filtering a list of documents in a field lookup

You can create a filtered list within a field lookup by running a custom loadscript from a schema subclass. The loadscript will filter the documents you specify every time someone loads a form using your custom schema subclasses.

To filter a list of documents in a field lookup:

1   Open a text editor such as Notepad.

2   Create a new loadscript file you want to use to specify what field lookup you
want to filter and the filter criteria. For example, to filter the list of ticket

categories to those relevant to the default company, enter the following load script:

```
function loadscript(msg)
{
   var explorer = personalize._getExplorer(
      msg.get(DOCEXPLORER_CONTEXT),
msg.get(DOCEXPLORER_INSTANCE) );
   var strAction = msg.get( DocExplorer.ACTION );

   ... // Examples 1 through 4

   // Example 5: Filtering field lookups
   if ( strAction == DocExplorer.ACTIONVALUE.LOOKUP )
   {
      var sRec = msg.get( "_lookuprecord" );
      if ( sRec == "category" )
      {
         // Filter category search by adding "Company" field
         var strQuery = msg.get( "query" );
         if ( strQuery.indexOf( "WHERE", 0 ) == -1 )
            strQuery += " WHERE ";
         else
            strQuery += " AND ";
            // Just a sample: change to filter by b. unit
            strQuery += " company=\"DEFAULT\"";

         msg.set( "query", strQuery );

         var msgCategories = archway.send( "sc", "query", msg
);

         var msgResponse = new Message( "fieldlookup" );
         msgResponse.add( msgCategories );
         return msgResponse;
      }
   }
   ...

   // Call default the onload script
   var script=msg.get(DocExplorer.LOADSCRIPT);
   if ( script != "" )
      msg = env.execute(script, msg);

   return msg;
}
```

The code executes when the action context is to perform a field lookup. This

is the case whenever the user presses a lookup icon in a DocExplorer.

3   Save the loadscript as an JS file in your custom scripts subfolder. For
example: `\custom\jscripts\tickets_by_contact.js`

Important: The load script file name must match the value listed in the
`<document>` loadscript attribute of your schema subclass.

4   Stop and restart your application server to pick up your schema changes.

Your new loadscript runs every time that someone accesses a form using
your schema subclass.

Adding data validation for document updates or inserts

You can create a server-side script to verify the validity of data before it is
updated or inserted to your back-end database. If the data is not valid, you can
have AssetCenter Web display an error message and return to the detail form for
the user to re-enter information. The loadscript validates the form data every
time someone submits the form.

To add data validation for document updates or inserts:

1   Open a text editor such as Notepad.

2   Create a new loadscript file you want to use to validate form entries. For example, to validate that the users do not enter the word "password" in the New Update field for a ticket, enter the following load script:

```
function loadscript(msg)
{
  var explorer = personalize._getExplorer(
     msg.get(DOCEXPLORER_CONTEXT),
msg.get(DOCEXPLORER_INSTANCE) );
  var strAction = msg.get( DocExplorer.ACTION );

   ... //Example 1

   // Example 2: Validate data before allowing an update
   if ( strAction == DocExplorer.ACTIONVALUE.UPDATE )
   {
     var s = msg.get( "NewUpdates" );
     var i = s.indexOf( "password", 0 );
     if ( i != -1 )
     {
        user.addMessage( "The word 'password' may not appear
in an update description. Please enter a different
description." );
        msg.set( DocExplorer.REDIRECT,
explorer.getFormNamePrefix() + "_detail.jsp" );
        return msg;
     }
   }

   ...
}
```

This validation function executes whenever the action context is of the type update. This is the case whenever a user presses the Update button to submit changes to a document.

3   Save the loadscript as an JS file in your custom scripts subfolder. For example: `\custom\jscripts\tickets_by_contact.js`

Important: The load script file name must match the value listed in the `<document>` loadscript attribute of your schema subclass.

4   Stop and restart your application server to pick up your schema changes.

Your new loadscript runs every time that someone accesses a form using your schema subclass.

Adding default values to a detail form

You can create a loadscript to add default values to a form based on the currently logged in user or other criteria. The loadscript will check for default values every time someone loads a form using your custom schema subclasses.

To add default values to a detail form:

1   Open a text editor such as Notepad.

2 Create a new loadscript file you want to use to add default values to your
form. For example, to add contact information for the currently logged in
user to the detail form, enter the following load script:

```
function loadscript(msg)
{
   var explorer = personalize._getExplorer(
      msg.get(DOCEXPLORER_CONTEXT),
msg.get(DOCEXPLORER_INSTANCE) );
   var strAction = msg.get( DocExplorer.ACTION );

   ... //Examples 1 and 2

   // Call default the onload script
   var script=msg.get(DocExplorer.LOADSCRIPT);
   if ( script != "" )
      msg = env.execute(script, msg);

   // Example 3: Adding default values to creation screen
   if ( strAction == PERSONALIZE_CREATE )
   {
      // Query for contact information
      var msgContact = this.getContact( user.get("_name") );

      // Augment initial document description
      var msgTicketByContact = msg.getMessage(
"tickets_by_contact" );
      if ( msgTicketByContact != null )
      {
         msgTicketByContact.set( "ContactName",
user.get("_name" ) );
         msgTicketByContact.remove( "Contact" );
         msgTicketByContact.add( msgContact );
      }
   }

   return msg;
}

function getContact( sName )
{
   var msgContact = archway.sendDocQuery( "sc", "SELECT * FROM
Contact
      WHERE Id='" + sName + "'", 0, 1 );
   msgContact = msgContact.getMessage( "Contact" );
   return msgContact;
}
```

This code executes when the action context is of type create. This is the case whenever the user accesses a document creation page.

3 Save the loadscript as an JS file in your custom scripts subfolder. For example: `\custom\jscripts\tickets_by_contact.js`

Important: The load script file name must match the value listed in the `<document>` loadscript attribute of your schema subclass.

4 Stop and restart your application server to pick up your schema changes.

Your new loadscript runs every time that someone accesses a form using your schema subclass.

Changing document data when meeting a particular condition

You can create a loadscript that checks for a particular condition and changes the data in a form before it is submitted to the AssetCenter Web back-end database. The loadscript will check for the condition you specify every time someone loads a form using your custom schema subclasses.

To change document data when meeting a particular condition:

1 Open a text editor such as Notepad.

2 Create a new loadscript file you want to use to specify what conditions result in document changes. For example, to change the ticket priority to 1 when user's department is set to Executive, enter the following load script:

```
function loadscript(msg)
{
   var explorer = personalize._getExplorer(
      msg.get(DOCEXPLORER_CONTEXT),
msg.get(DOCEXPLORER_INSTANCE) );
   var strAction = msg.get( DocExplorer.ACTION );

   ... //Examples 1 through 3

   // Example 4: Modify data before ticket creation
   if ( strForm.indexOf( "_new" ) != -1 )
   {
      // Set ticket priority for some users
      var msgContact = this.getContact( msg.get("ContactName")
);
      var sDept = msgContact.get( "Department" );
      var sPrio = "3";
      if ( sDept == "Executive" )
         sPrio = "1";
      msg.set( "tickets_by_contact/Priority", sPrio );
   }

   // Call default the onload script from the problem schema
   var script=msg.get(DocExplorer.LOADSCRIPT);
   if ( script != "" )
      msg = env.execute(script, msg);

   return msg;
}
```

This code executes when the action context is of type create. This is the case whenever the user accesses a document creation page.

3 Save the loadscript as an JS file in your custom scripts subfolder. For example: `\custom\jscripts\tickets_by_contact.js`

> **Important:** The load script file name must match the value listed in the `<document>` loadscript attribute of your schema subclass.

4 Stop and restart your application server to pick up your schema changes.

Your new loadscript runs every time that someone accesses a form using your schema subclass.

Schema elements and attributes

All schemas use a standard set of XML elements and attributes that the Archway Document Manager recognizes. The following sections describe the XML elements and associated attributes that you can use to create valid schemas.

<?xml>

The `<?xml>` element is the standard XML namespace identifier. This element should always include the version attribute. All schemas require that this be the first element listed.

<schema>

The `<schema>` element is a required element of all schemas. The `<schema>` element functions as a container for the logical and physical mappings. The `<schema>` element does not have any attributes.

<documents>

Two sets of `<documents>` elements are required for each schema. One set of `<documents>` elements is the container for the logical mappings and the other set of `<documents>` elements is the container for the physical mappings.

Use in logical mapping

All schemas require one `<documents>` element where the `name` attribute has the value `name="base"`. When this element has this `name` value, it becomes the container for the logical mappings.

Required attributes

Attribute	Description
name	This attribute identifies the <documents> element container used by the logical mappings. This attribute must have the value name="base".

Optional attributes

Attribute	Description
none	There are no optional attributes for the logical mapping portion of the schema.

```
<?xml version="1.0"?>
<schema>

<documents name="base">
   ...
</documents>

...
```

Use in physical mapping

All schemas require at least one <documents> element where the name attribute has the value of an adapter name such as name="ac". You can add one <documents> element for each adapter you want to provide physical mappings for. You can also support multiple versions of the same adapter if you use the version attribute.

Required attributes

Attribute	Description
name	This attribute determines what adapter the schema uses to make connections to the back-end database. The value of this attribute must be an adapter name such as name="ac".

Optional attributes

Attribute	Description
version	This attribute determines what version of the back-end database is required to use the physical mappings defined in this container. The value of this attribute must be a number recognized by the adapter.

```
<?xml version="1.0"?>
<schema>

...

<documents name="ac" version="3">
   ...
</documents>

<documents name="ac" version="4">
   ...
</documents>

...
```

The Archway Document Manager uses the following rules to match the back-end database to the version listed in this attribute:

- If the <documents> element has *no* version attribute, then the Archway Document Manager accepts the physical mappings in this element if it cannot find another matching value.

- If the <documents> element has a version attribute value *greater* than the version number of the back-end database, then the Archway Document Manager ignores the physical mappings in this element.

- If the <documents> element has a version attribute value *less* than the version number of the back-end database, then the Archway Document Manager accepts the physical mappings in this element if it cannot find a higher matching value.

- If the <documents> element has a version attribute value *equal* to the version number of the back-end database, then the Archway Document Manager accepts the physical mappings in this element.

\<document>

You must add at least two sets of \<document> elements to create a valid schema – one set for the logical mappings and another set for the physical mappings. You can add additional \<document> elements in the physical mapping section if you want to support multiple adapters or multiple versions of the same back-end database.

Use in logical mapping

The logical mapping section uses the \<document> elements as a container for the XML document that the Archway Document Manager produces. All XML elements produced by this schema will be child elements of the \<document> element.

Required attributes

Attribute	Description
name	This attribute determines what XML element the Archway Document Manager generates as the top-level element in any generated document using this schema. The value of this attribute must match the file name of the schema (*without* the .xml extension).

Optional attributes

Attribute	Description
ACLcreate	This attribute determines the default access control list for DocExplorer forms that use this schema. The value of this attribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see a **Create** button in DocExplorer forms that use this schema.
ACLdelete	This attribute determines the default access control list for DocExplorer forms that use this schema. The value of this attrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see a **Delete** button in DocExplorer forms that use this schema.
ACLupdate	This attribute determines the default access control list for DocExplorer forms that use this schema. The value of this attrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will be able to edit fields in DocExplorer detail forms that use this schema.

Attribute	Description
create	This attribute determines if a subdocument using this element is visible in DocExplorer *create* forms. The value of this attribute must be either true or false. Set the value to `create="true"` if you want this subdocument to be available on DocExplorer create forms. Set the value to `create="false"` if you want to prevent this subdocument from being available on DocExplorer create forms.
detail	This attribute determines if a subdocument using this element is visible in DocExplorer *detail* forms. The value of this attribute must be either true or false. Set the value to `detail="true"` if you want this subdocument to be available on DocExplorer detail forms. Set the value to `detail="false"` if you want to prevent this subdocument from being available on DocExplorer detail forms.
docname	This attribute defines the external schema that you want the Archway Document Manager to use to create a subdocument. The value of this attribute must match the file name of the schema (*without* the `.xml` extension) that you want to use for the subdocument. You only need this attribute if you want to create a subdocument using an another schema.
label	This attribute determines what name the schema has in DocExplorer forms that use this schema. The value of this attribute can be any text string. Typically, you will want to set this value to a user-friendly name describing the content of the schema.
list	This attribute determines if a subdocument using this element is visible in DocExplorer *list* forms. The value of this attribute must be either true or false. Set the value to `list="true"` if you want this subdocument to be available on DocExplorer list forms. Set the value to `search="false"` if you want to prevent this subdocument from being available on DocExplorer list forms.
loadscript	This attribute determines what ECMAScript runs when this schema is used in a DocExplorer form. The value of this attribute must be the Peregrine Studio name of the ECMAScript you want to run. You can use this script to load additional data for use by DocExplorer forms. This script uses the same XML message input as the form onload script. See *Document Schema Extensions* for examples of loadscripts.
preexplorer	This attribute determines what ECMAScript runs when this schema is used in a DocExplorer form. The value of this attribute must be the Peregrine Studio name of the ECMAScript you want to run. You can use this script to make formatting changes to the XML message rendered by DocExplorer forms. See your AssetCenter Web deployment for examples of pre-explorer scripts. Pre-explorer scripts are located at the following path: `<application server>\oaa\WEB-INF\apps\<package>\jscript\preexplorer`

Attribute	Description
search	This attribute determines if a subdocument using this element is visible in DocExplorer *search* forms. The value of this attribute must be either true or false. Set the value to `search="true"` if you want this subdocument to be available on DocExplorer search forms. Set the value to `search="false"` if you want to prevent this subdocument from being available on DocExplorer search forms.
subtypeprop	This attribute determines whether this element inherits the attribute properties of the parent `<collection>` element. The value of this attribute must be `inherit` if you use the attribute at all. If you want this element to inherit the attribute properties set the value to `subtypeprop="inherit"`. If you want to specify the attribute properties for this element, do not include a `subtypeprop` attribute.

Use in physical mapping

The physical mapping section uses the `<document>` elements to define the SQL name of the back-end database table.

Required attributes

Attribute	Description
name	This attribute determines what XML element the Archway Document Manager matches to a back-end database table. The value of this attribute must match the file name of the schema (*without* the `.xml` extension).
table	This attribute identifies the table in the back-end database that the schema uses.The value of this attribute must be the SQL name of the table you want to use for source data. Each `<document>` element can only have one `table` attribute. To use data from other tables, you can create subdocuments within your schema.

Optional attributes

Attribute	Description
field	This attribute identifies the field in the back-end database that you want the schema to use for document queries. The value of this attribute must be the SQL name of the field you want to use for the data source. You only need this attribute if you want to create a subdocument within your schema. You can also set this attribute to _null if there is no physical mapping for this document in your back-end database.
joinfield	This attribute identifies the field in the back-end database that you want the schema to use to query for additional information in another schema or table. The value of this attribute must be the SQL name of the field you want to use for the source data. You only need this attribute if you want to create a subdocument within your schema. The joinfield attribute defines what field will be the selection criteria in a SQL WHERE clause. The SQL equivalent of the joinfield is:
	SELECT <field> FROM <external table> WHERE <joinfield>=<joinvalue>
	If you do not provide a joinfield value, then the Archway Document Manager uses the field listed for the <attribute name="Id"> element as the joinfield.
joinvalue	This attribute identifies the <attribute> element that has the value you want to use to query for additional information in another schema or table. The value of this attribute must be the name of an <attribute> element in the current schema. You only need this attribute if you want to create a subdocument within your schema. The joinvalue attribute defines what value a field must have in a SQL WHERE clause. The SQL equivalent of the joinvalue is:
	SELECT <field> FROM <external table> WHERE <joinfield>=<joinvalue>
	If you do not provide a joinvalue value, then the Archway Document Manager uses the value returned for the <attribute name="Id"> element as the joinvalue.
link	This attribute identifies the field in the back-end database that you want the schema to use to query for additional information in a table with lookup or link fields. The value of this attribute must be the SQL name of the field you want to use for the source data. You only need this attribute if you want to create a subdocument within your schema. In most cases, the link attribute is the same as the joinfield attribute. This value will only be different if the SQL name of the link field in the source table is different from the SQL name from the target field in the target table.

Attribute	Description
preprocess	This attribute determines what ECMAScript runs *before* the Archway Document Manager connects to the back-end database. The value of this attribute must be the Peregrine Studio name of the ECMAScript you want to run. You can use this script to format the request sent to the back-end database. For example, you can add additional SQL commands or validate that all required fields are listed in the request. See your AssetCenter Web deployment for examples of pre-process scripts. Pre-process scripts are located at the following path: `<application server>\oaa\WEB-INF\apps\ <package>\jscript\schema`
postprocess	This attribute determines what ECMAScript runs *after* the Archway Document Manager receives a response from the back-end database. The value of this attribute must be the Peregrine Studio name of the ECMAScript you want to run. You can use this script to format the response sent from the back-end database. For example, you can sort the data by a particular criteria or return an error message if no records are found. See your AssetCenter Web deployment for examples of post-process scripts. Post-process scripts are located at the following path: `<application server>\oaa\WEB-INF\apps\ <package>\jscript\schema`

<attribute>

You must add at least two sets of `<attribute>` elements to create a valid schema – one set for the logical mappings and another set for the physical mappings.

Use in logical mapping

The logical mapping sections use the `<attribute>` elements to create an XML element in any document message built from this schema.

Required attributes

Attribute	Description
name	This attributes determines the XML tag that the Archway Document Manager generates when it uses the schema. The value of this attribute can be any string value. For example, if you set the value to name="contact" then the Archway Document Manager creates a <contact> XML tag. You must define at least one <attribute> element where the name attribute has the value name="Id". This <attribute> element is required to uniquely identify each record returned by a schema query.
type	This attribute determines what data format the elements uses as well as how AssetCenter Web renders the data in the user interface. The value of this attribute must be one of the following strings. **Note:** The Archway Document Manager does not validate that the contents of an element matches the type attribute listed for it.
attachment	This element is a path and file name to an attachment. AssetCenter Web renders this element as a collection of attachment controls.
boolean	This element is a true or false string. AssetCenter Web renders this element as a check box.
date	This element is a date listing. AssetCenter Web renders this element as a date edit control that includes a popup calendar.
datetime	This element is a combined date and time listing. AssetCenter Web renders this element as a time edit control.
id	This element is a number that uniquely describes a back-end database record. AssetCenter Web renders this element as a single-line edit field.
image	This element is an image. AssetCenter Web renders this element as an imagefield.
link	This element is a subdocument described elsewhere in the schema. AssetCenter Web renders this element as a lookup field.
memo	This element is a text string. AssetCenter Web renders this element as a multi-line edit box.
money	This element is a currency amount. AssetCenter Web renders this element as a money field that includes a currency selection tool.
number	This element is an integer. AssetCenter Web renders this element as an editfield with spinner buttons.
preload	This element is an executable script. AssetCenter Web runs the script listed in this element.
string	This element is text. AssetCenter Web renders this element as an editfield.

Attribute	Description
`time`	This element is a time listing. AssetCenter Web renders this element as a time edit control.
`url`	This element is a Web site address. AssetCenter Web renders this element as an HREF link icon.

Optional attributes

Attribute	Description
`access`	This attribute determines whether the field described by this element accepts updates or inserts in the back-end database or whether it is a read-only field. The value of this attribute must be either r or null. Set the value to `access="r"` if you want to make this element read-only. Clear the value or remove the attribute if you want to enable updates and inserts to this field.
`ACLcreate`	This attribute determines the default access control list for DocExplorer forms that use this element. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see this element in DocExplorer *create* forms that use this schema.
`ACLdetail`	This attribute determines the default access control list for DocExplorer forms that use this element. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see this element in DocExplorer *detail* forms that use this schema.
`ACLlist`	This attribute determines the default access control list for DocExplorer forms that use this element. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see this element in DocExplorer *list* forms that use this schema.
`ACLsearch`	This attribute determines the default access control list for DocExplorer forms that use this element. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see this element in DocExplorer *search* forms that use this schema.
`create`	This attribute determines if the element is visible in DocExplorer *create* forms. The value of this attribute must be either true or false. Set the value to `create="true"` if you want this field to be available on DocExplorer create forms. Set the value to `create="false"` if you want to prevent this field from being available on DocExplorer create forms.
`detail`	The value of this attribute must be either true or false. Set the value to `detail="true"` if you want this field to be available on DocExplorer detail forms. Set the value to `detail="false"` if you want to prevent this field from being available on DocExplorer detail forms.

Attribute	Description
enum	For enum attributes that return a string value rather than a numeric index:
	Create the enum values with a numeric storage value, and the value name equal to the actual value to be stored in the backend.
	Define your field in the schema with the following attribute values - type="enum" exttype="<enum name>" localize="true" valueprop="name".
label	This attribute determines what name the element has in DocExplorer Available Field list. The value of this attribute can be any text string. Typically, you will want to set this value to a user-friendly name describing the content of the field.
list	This attribute determines if the element is visible in DocExplorer list forms. The value of this attribute must be either true or false. Set the value to list="true" if you want this field to be available on DocExplorer list forms. Set the value to search="false" if you want to prevent this field from being available on DocExplorer list forms.
required	This attribute determines if this element requires a value in order to insert or update a record in the back-end database. The value of this attribute must be either true or false. Set the value to required="true" if you want to make the element a required input field when it is added to DocExplorer forms.
search	This attribute determines if the element is visible in DocExplorer *search* forms. The value of this attribute must be either true or false. Set the value to search="true" if you want this field to be available on DocExplorer search forms. Set the value to search="false" if you want to prevent this field from being available on DocExplorer search forms.

Use in physical mapping

The physical mapping sections use the `<attribute>` elements to define the fields in the back-end database that map to each logical mapping.

Required attributes

Attribute	Description
`name`	This attributes determines the XML tag in which the Archway Document Manager places query results. The value of this attribute must match an element defined in the logical mapping section.
`field`	This attribute identifies the field in the back-end database that you want the schema to use for document queries. The value of this attribute must be the SQL name of the field you want to use for the data source. You can also set this attribute to `_null` if there is no physical mapping for this field in your back-end database.

Optional attributes

Attribute	Description
`link`	This attribute identifies a lookup or link value to another table. The value of this attribute must be the SQL name of the link. You will only need this attribute if you want to query information from a field in one table that links to another field in a linked table. The link attribute defines what field is the selection criteria in a SQL WHERE clause. The SQL equivalent of the `link` is: `SELECT <linkfield> FROM <linktable> WHERE <link>=<field>`
`linkfield`	This attribute identifies the target field called by a lookup or link value to another table. The value of this attribute must be the SQL name of the target field. You will only need this attribute if you want to query information from a field in one table that links to another field in a linked table. The `linkfield` attribute defines what field is selected. The SQL equivalent of the `link` is: `SELECT <linkfield> FROM <linktable> WHERE <link>=<field>`

Attribute	Description
linkkey	This attribute identifies the field, lookup, or link that connects two fields in linked tables. The value of this attribute must be the SQL name of the linking field. You will only need this attribute if you want to query information from a field in one table that links to another field in a linked table. The linkkey attribute defines what field is selected. The SQL equivalent of the link is: SELECT <linkfield> FROM <linktable> WHERE <linkkey>=<field> If you do not define a linkkey value, then the Archway Document Manager uses the link attribute as the linkkey.
linktable	This attribute identifies the target table called by a lookup or link value. The value of this attribute must be the SQL name of the target table. You will only need this attribute if you want to query information from a field in one table that links to another field in a linked table. The linktable attribute defines what table is named in a SQL FROM clause. The SQL equivalent of the linktable is: SELECT <linkfield> FROM <linktable> WHERE <link>=<field>
linktype	This attribute defines how the Archway Document Manager performs document inserts and updates. The value of this attribute must be either soft or hard:
soft	The Archway Document Manager queries the back-end database using the locations listed in the linktable and linkfield attributes, and sets the link attribute to the value to the query result.
hard	The Archway Document Manager creates a new record in the back-end database at the location listed in the linktable and linkfield attributes. The Archway Document Manager retrieves the linkkey value for the new record and saves it in the field listed in the link attribute.
	Note: If you do not specify a linktype value, then it defaults to soft. You will only need this attribute if you want to query information from a field in one table that links to another field in a linked table.

\<collection>

This is an optional element that you can use to create subdocuments where more than one item can be returned for the document you query. For example, you can create a set of <collection> elements to query for all the assets that a particular user has. In database terminology, a <collection> element returns the records from an intersection table. You must add one set of <collection> elements for each multiple item subdocument you want to create.

Use in logical mapping

The logical mapping section uses the `<collection>` elements to create the XML elements that the subdocuments use.

Required attributes

Attribute	Description
name	This attribute determines what XML element the Archway Document Manager generates as the top-level element in any generated document using this schema. The value of this attribute must match the file name of the schema (*without* the `.xml` extension) that the subdocument uses.

Optional attributes

Attribute	Description
ACLcreate	This attribute determines the default access control list for DocExplorer forms that use this subdocument. The value of this attribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see a **Create** button in DocExplorer forms that use this schema.
ACLdelete	This attribute determines the default access control list for DocExplorer forms that use this subdocument. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will see a **Delete** button in DocExplorer forms that use this schema.
ACLupdate	This attribute determines the default access control list for DocExplorer forms that use this subdocument. The value of this atrribute must be a capability word. Users who meet or exceed the capability word listed in this attribute will be able to edit fields in DocExplorer detail forms that use this schema.
create	This attribute determines if a subdocument using this element is visible in DocExplorer *create* forms. The value of this attribute must be either true or false. Set the value to `create="true"` if you want this subdocument to be available on DocExplorer create forms. Set the value to `create="false"` if you want to prevent this subdocument from being available on DocExplorer create forms.

Attribute	Description
detail	This attribute determines if a subdocument using this element is visible in DocExplorer *detail* forms. The value of this attribute must be either true or false. Set the value to detail="true" if you want this subdocument to be available on DocExplorer detail forms. Set the value to detail="false" if you want to prevent this subdocument from being available on DocExplorer detail forms.
label	This attribute determines what name the subdocument has in DocExplorer forms that use this schema. The value of this attribute can be any text string. Typically, you will want to set this value to a user-friendly name describing the content of the schema.
list	This attribute determines if a subdocument using this element is visible in DocExplorer list forms. The value of this attribute must be either true or false. Set the value to list="true" if you want this subdocument to be available on DocExplorer list forms. Set the value to search="false" if you want to prevent this subdocument from being available on DocExplorer list forms.
search	This attribute determines if a subdocument using this element is visible in DocExplorer *search* forms. The value of this attribute must be either true or false. Set the value to search="true" if you want this subdocument to be available on DocExplorer search forms. Set the value to search="false" if you want to prevent this subdocument from being available on DocExplorer search forms.

Use in physical mapping

The physical mapping section uses the <collection> elements to define the SQL name of the back-end database table.

Required attributes

Attribute	Description
name	This attribute determines what XML element the Archway Document Manager matches to a back-end database table. The value of this attribute must match the file name of the schema (*without* the .xml extension).

Optional attributes

Attribute	Description
none	There are no optional attributes for the physical mapping portion of a <collection> element.

Documents

The Archway Document Manager uses schemas to create documents, which are XML messages created from the following components.

Component	Description
Schema logical definitions	The schema logical definitions determine what XML elements make up the generated document.
Return values of database queries	The Archway Document Manager uses the schema physical mappings to create database queries. The return values of these queries determine the content of the elements and attributes of the generated document.
ECMAScript formatting	ECMAScripts can modify a document before and after any queries have been made to the back-end database.

The final output of these three processes is an XML document that the Archway Document Manager renders as HTML in the AssetCenter Web interface.

You can see the raw AssetCenter Web XML documents by enabling the **Show form information** option from the Administration settings. The form information window displays the following document information.

Tab	Description
Script Input	This tab displays the document submitted to the current form from the output of a previous form. For example, a list form displays the output of a prior search form. This document is passed to the form onload script as an input parameter.
Script Output	This tab displays the document generated by the output of the current form's onload script. Typically, each onload script invokes a schema that queries the back-end database for relevant information. For example, a contract form will invoke a database query through the contract schema.
PreXSL	This tab displays the document after the Archway servlet has processed the document and prepared it to be rendered by the client-side browser.

Subdocuments

Each AssetCenter Web form typically maps to one schema, which in turn maps to one table in the back-end database. In order to collect and represent data from multiple schema and database sources, you must create subdocuments.

Subdocuments are XML messages added to the current document that query additional schemas and tables. You can create subdocuments in one of two ways:

- You can add a new `<document>` element inside an existing `<document>` element if the result of the query will be *one and only one* subdocument.
- You can add a `<collection>` element inside an existing `<document>` element if the result of the query will be a collection of *one or more* subdocuments.

The following sections examples of each method.

Creating subdocuments with the <document> element

Each `<document>` element is intended to return one subdocument, that is, one record set. For example, you can create subdocument to query for the contact name for a specific contract, but each contract should only have one contact name.

Schema

The following schema segment illustrates how to add a subdocument using the `<document>` element.

```
<documents name="base">
   <document name="Request" label="Request"...>
      <attribute name="Id" type="id".../>
      <attribute name="Number" type="string" label="Number".../>
      <attribute name="Purpose" type="string"
label="Purpose".../>
      ...
      <document name="EndUser" docname="Employee" label="End
User"/>
      ...
   <document>
<documents>

<documents name="ac" version="4">
   <document name="Request" table="amRequest"...>
      <attribute name="Id" field="lReqId"/>
      <attribute name="Number" field="ReqNumber"/>
      <attribute name="Purpose" field="ReqPurpose"/>
...
      <document name="EndUser" docname="Employee"
table="amEmplDept"
       field="lUserId" link="lUserId" joinfield="lEmplDeptId"
       joinvalue="EndUserId"/>
      ...
   <document>
<documents>
```

XML output

The Archway Document Manager produces an XML document with the
following structure. You can view such documents from the Script Input and
Script Output tabs of the Form Information window. The values stored in the
XML elements vary depending on the actual user record you select.

```
<Request>
   <Id>32097</Id>
   <Number>REQ000042</Number>
   <Purpose>Purpose 1</Purpose>
   ...
   <EndUserId>15630</EndUserId>
   ...
</Request>
```

Creating subdocuments with the <collection> element

Each <collection> element is intended to return more than one subdocument or record set. For example, you can create a query to return all the contracts belonging to a particular contact.

Schema

The following schema segment illustrates how to add a subdocument using the <collection> element.

```
<documents name="base">
   <document name="Request" label="Request"...>
      <attribute name="Id" type="id".../>
      <attribute name="Number" type="string" label="Number".../>
      <attribute name="Purpose" type="string"
label="Purpose".../>
      ...
      <collection     name="RequestLines" label="Composition">
         <document       name="RequestLine"/>
      </collection>
      ...
   <document>
<documents>

<documents name="ac" version="4">
   <document name="Request" table="amRequest"...>
      <attribute name="Id" field="lReqId"/>
      <attribute name="Number" field="ReqNumber"/>
      <attribute name="Purpose" field="ReqPurpose"/>
      ...
      <!-- No physical mapping for the RequestLines collection.
-->
      ...
   <document>
<documents>
```

```
<documents name="base">
   <document name="RequestLine" label="Request Line"...>
      <attribute name="Id" type="id" search="false" list="false"
       detail="false" create="false" />
      ...
      <collection name="RequestLines" label="Composition"
detail="true"
       create="true">
         <document name="RequestLine" table="_null"/>
      </collection>
      ...
   <document>
<documents>

<documents name="ac" version="4.0">
   <document name="RequestLine" table="amReqLine"...>
      <attribute name="Id" field="lReqLineId" />
      ...
      <collection    name="RequestLines" label="Composition">
         <document    name="RequestLine" table="_null"
          joinfield="lParentId" />
      </collection>
      ...
   <document>
<documents>
```

XML output

The Archway Document Manager produces an XML document with the
following structure. You can view such documents from the Script Input and
Script Output tabs of the Form Information window. The values stored in the
XML elements vary depending on the actual user record you select.

```
<Request>
   <Id>32098</Id>
   <Number>REQ000043</Number>
   <Purpose>Purpose 2</Purpose>
   ...
   <RequestLines _count="-1" _countFound="3" _more="0"
_start="0">
      <RequestLine>
         <Id>32100</Id>
         <RequestId>32098</RequestId>
         <RequestNumber>REQ000043</RequestNumber>
         ...
      </RequestLine>

      <RequestLine>
         <Id>32101</Id>
         <RequestId>32098</RequestId>
         <RequestNumber>REQ000043</RequestNumber>
         ...
      </RequestLine>

      <RequestLine>
         <Id>32102</Id>
         <RequestId>32098</RequestId>
         <RequestNumber>REQ000043</RequestNumber>
         ...
      </RequestLine>
   </RequestLines>
</Request>
```

6
CHAPTER

Modifying the Request Type and Item Type Selection Menus

This section describes how to customize forms used in AssetCenter Web that cannot be personalized from the browser (the wrench icon does not show up for these pages), but that can be configured through XML files.

This section includes:

- Configuring the hierarchical menu component on page 137

Configuring the hierarchical menu component

You can configure the menu forms used in through XML files.

General features of the menu component

The following graphic shows the menu parts that you can configure.

The configuration file allows setting up:

- The **Title** of the form.
- The **Instructions** presented at the top of the form.
- The **Menu Entries** available on the form, each represented by an icon and a label.

Each menu entry:

- Can be a final selection, and when the user clicks on it, the selected option is passed to the application.

Note: You can also configure a final selection node to redirect a given URL to a specific form in AssetCenter Web, or to another web application, or an external web site.

- Can lead to a submenu, a new form that has its own title, instructions, and options.

The following submenu has text below the instructions that indicates where the form originates.

All or part of the menu can be generated dynamically from the data contained in the database.

Syntax of a menu configuration file

A menu configuration file is an XML file. Its syntax is described in the W3C schema (XML schema): `WEB-INF\etc\treemenu\treemenu.xsd`.

The root element: WizardMenu

A WizardMenu element is always at the root of the XML file. It usually has two attributes that describe which W3C schema describes the file syntax:

```
<WizardMenu xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="menu.xsd">
```

Directly under the WizardMenu element, its subelements describe the menu form.

Subelement	Description
Id	Optional; contains a value identifying this element. This is the value passed to the application when the menu has no Answers element or when the Answers element is empty.
Title	Optional; contains the text displayed for the form title.

Subelement	Description
Title_ids	Optional; contains the string id representing the text displayed for the form title. This value is expressed as module,stringname where module corresponds to the file name containing the string, and stringname is the id of the string in this file.
	If **Title** is specified, it has precedence over **Title_ids**, and all users, regardless of the locale they choose when they log in, view the same exact text.
Instructions	Optional; contains the text displayed for the instructions at the top of the form.
Instructions_ids	Optional; contains the string id representing the text displayed for the instructions at the top of the form.
	Note: If Instructions is specified, it has precedence over Instructions_ids, and all users, regardless of the locale they choose when they log in, view the same exact text.
Access	Optional; this element contains one capability word, such as getit.requester or getit.service. Only users with the given capability word can see this menu entry.
ColumnCount	Optional; this element specifies the number of columns in which the menu entry should be arranged on the screen. The default value is set by the application using the menu tree.
Answers	Optional; describes the options (menu entries) available on the form. When the Answers element is not provided, or when it does not list any option, the form returns the WizardMenu Id, when provided.

The list of menu entries: the Answers element

The Answers element describes the options available in a menu. Each subelement of the Answers element corresponds to one or more options. Each type of element can be used more than one time in a given Answers element. The supported subelements types are:

Subelement	Description
WizardTarget	Optional; describes one option available in the menu. This element does not lead to a submenu. When the user selects this option, the selected WizardTarget's Id element is passed to the application.
WizardMenu	Optional; describes one option that, when selected, leads to a submenu presenting more options to the end user.
DynamicAnswers	Optional; describes a set of options that can be retrieved dynamically from a database.

The simple selection option: the WizardTarget element

Option	Description
Id	Required element that must be unique among the siblings of the Answers element.
Title	Optional; contains the text displayed for the menu entry.
Title_ids	Optional; contains the string id representing the text displayed for the menu entry. This value is expressed as module,stringname where module corresponds to the file name containing the string, and stringname is the id of the string in this file. **Note:** **Title** or **Title_ids** must be specified. If **Title** is specified, it has precedence over **Title_ids**, and all users, regardless of the locale they choose when they log in, view the same exact text.
Instructions	Optional; contains the text displayed in a tooltip when the user hovers the mouse cursor over the entry.
Instructions_ids	Optional; contains the string id representing the text displayed in a tooltip when the user hovers the mouse cursor over the entry. This value is expressed as module,stringname where module corresponds to the file name containing the string, and stringname is the id of the string in this file. **Note:** If **Instructions** is specified, it has precedence over **Instructions_ids**, and all users, regardless of the locale they choose when they log in, view the same exact text.
Image	Optional; this element is the path to the image that is displayed on the screen in front of the text for this menu entry. The value is a path to the icon, relative to the skin directory (for example, `icons/oaa_assets.gif`).
Access	Optional; this element contains one capability word, such as getit.requester or getit.service. Only users with the given capability word can see this menu entry.
TargetForm	Optional; name of the form the application is redirected to when the user clicks on the menu entry. The value is expressed as modulename.activityname.formname, where modulename is the name of the module where the target form is located, activityname is the name of the activity where the file is located, and formname the name of the form itself. The value can also be expressed as activityname.formname, in which case the module is implicitly the current module, or just formname, in **which** case the form is searched in the current module and activity.

Option	Description
TargetURL	Optional; this element contains the URL of the form to go to when the user clicks on the menu entry. Make sure to start this URL with http:// when redirecting to a web server other than the current web server. Instead of redirecting to a page, this menu entry can be used to retrieve documents stored on a server. All usual protocols can be used (http, https, ftp). **Note:** At most, only one **TargetForm** and **TargetURL** can be present in a given WizardTarget element.
TargetAddNoParams	Optional; this Boolean element, when set to true, prevents automatically passing parameters to the **TargetURL** or the **TargetForm** when the menu entry is selected.
TargetParams	Optional; this element is the ampersand-separated list of parameters to add to the **TargetURL** or to pass to the **TargetForm**. If **TargetAddNoParams** is not set or set to false, these parameters are passed in addition to the parameters already added automatically.
ContextFilter	Optional; this element represents a filter on the context data. This menu entry is displayed only if the context data matches the requirement of the filter. The context data depends on the application, but contains at least the user login name.

The submenu option: the WizardMenu element

Do not confuse this element with the root WizardMenu element. It is similar in structure, but has also more options. It is represented as a single entry in the menu. Clicking on it leads to a submenu.

Option	Description
Id:	Required element that must be unique among the siblings of the Answers element.
Title:	Optional; contains the text displayed in for the menu entry. This text becomes the title of the submenu form.
Title_ids	Optional; contains the string id representing the text displayed for the menu entry. This text becomes the title of the submenu form.This value is expressed as module,stringname where module corresponds to the file name containing the string, and stringname is the id of the string in this file. **Note:** **Title** or **Title_ids** must be specified. If Title is specified, it has precedence over **Title_ids**, and all users, regardless of the locale they choose when they log in, view the same exact text.

Option	Description
Instructions	Optional; contains the text displayed in a tooltip when the user hovers the mouse cursor over the entry. This text becomes the instructions of the submenu form.
Instructions_ids	Optional; contains the string id representing the text displayed in a tooltip when the user hovers the mouse cursor over the entry. This text becomes the instructions of the submenu form. This value is expressed as module,stringname where module corresponds to the file name containing the string, and stringname is the id of the string in this file. **Note:** If **Instructions** is specified, it has precedence over **Instructions_ids**, and all users, regardless of the locale they choose when they log in, view the same exact text.
Image	Optional; this element is the path to the image that is displayed on the screen in front of the text for this menu entry. The value is a path to the icon, relative to the skin directory (for example, `icons/oaa_assets.gif`).
Access	Optional; this element contains one capability word, such as getit.requester or getit.service. Only users with the given capability word can see this menu entry.
TargetForm	Optional; name of the form the application is redirected to when the user clicks on the menu entry and that there are no submenu entries. The value is expressed as modulename.activityname.formname, where modulename is the name of the module where the target form is located, activityname is the name of the activity where the file is located, and formname the name of the form itself. The value can also be expressed as activityname.formname, in which case the module is implicitly the current module, or just formname, in which case the form is searched in the current module and activity.
TargetURL	Optional; this element contains the URL of the form to go to when the user clicks on the menu entry and that there are no submenu entries. Make sure to start this URL with http:// when redirecting to a web server other than the current web server. Instead of redirecting to a page, this menu entry can be used to retrieve documents stored on a server. All usual protocols can be used (http, https, ftp). **Note:** At most, only one **TargetForm** and **TargetURL** can be present in a given WizardMenu element.
TargetAddNoParams	Optional; this Boolean element, when set to true, prevents automatically passing parameters to the **TargetURL** or the **TargetForm** when the menu entry is selected and there are no submenu entries.

Option	Description
TargetParams	Optional; this element is the ampersand separated list of parameters to add to the TargetURL or to pass to the TargetForm. If TargetAddNoParams is not set or set to false, these parameters are passed in addition to the parameters already added automatically.
ContextFilter	Optional; this element represents a filter on the context data. This menu entry is displayed only if the context data matches the requirement of the filter. The context data depends on the application but contains at least the user login name.
ColumnCount	Optional; this element specifies the number of columns in which the menu entry should be arranged on the screen. The default value is set by the application using the menu tree.
Answers	Optional; describes the options (menu entries) available on the submenu form. When the Answers element is not provided, or when it does not list any option, the information for this form is used.

Dynamic menu entries: the DynamicAnswers element

Option	Description
Target	Required; name of the back-end system (for example, ac or sc) where the menu data is stored.
Document	Required; name of the schema that retrieves the menu data. The schema must map at least an Id and a Title, but it can also map any element available in a WizardTarget or WizardMenu.
Image	Optional; this is the path to the image that is displayed on the screen in front of the text for this menu entry when no image is retrieved from the database. The value is a path to the icon, relative to a skin directory (for example, `icons/catbundle.gif`).
Access	Optional; this element contains one capability word, such as getit.requester or getit.service. This DynamicAnswers element is available only for users with the given capability word.

Option	Description
HasSubMenu	Optional; when set to false, all entries returned are considered as final selection entries. When not set, or set to true, and when a user selects one of these entries, the program tries to build a menu with the content of the Answers element, or if there is no Answers element, it sets a **ParentId** parameter to the selected menu **Id** and re-executes the database search using the current DynamicAnswers element.
QueryParam	Optional; this element represents the search parameters that are used to filter the list of menu entries. The actual search parameters that can be used depend on the schema defined in the Document element. This element contains one or more subelements. The name of one of these subelements is the attribute name (which can be found in the schema), and the value set is used in the query that retrieves the menu entry information.
Answers	Optional; describes the options (menu entries) available on the submenu form. Follows the syntax for an Answers element described above.

Note: An alternative to providing a Target and a Document element is to provide a Script element that represents an ECMA script function name. The Target and Document elements can optionally be specified, if the function needs them. The script is getting passed the current node definition, plus the ParentId corresponding to the last menu entry that was clicked.

CHAPTER 7

Tree Menu Enhancements

The tree menu includes several enhancements:

- Decouple page title and instructions from option label and description.
- Clickable Trace widget, showing the selections made, and allowing the user to go back to a previous levels.
- Per page, allow viewing the options as icons (as before) or in a list view.
- Per page, possibility to show a **Done** button that returns the current selection bypassing the sub options.
- Wizard-like behavior: an application can set up the tree menu to collect data as the options are selected. This allows collecting more than one value where the tree menu was only able to select the last Id.
- Default values for dynamic entries.

General enhancements

The following sections describe some of the general tree menu enhancements.

Title and instructions vs. label and description

In AssetCenter Web 4.1, the tree menu used two strings per page: **Title** and **Instructions**. The following table describes how they were used.

Type	Title	Instructions
menu page	represented the form title	showed as form instructions (top of the form)
menu entry	used for the icon's caption	used to display a long description in a tool tip

All this worked well as long as you had only one page. But if you had more than one level, on the second level, suddenly, the title was changed to the selected item's caption, and the instructions replaced with the selected items description, rather than with real instructions on what to do on that second level.

In addition, some customers wanted to use the tree as a decision tree, where on a page the instructions would describe a question, and the options would represent answers. One answer would lead to the next level, which would present a new question and a new set of answers. This did not work well with the AssetCenter Web 4.1 model.

AssetCenter Web 4.2 separates the title and instructions for a page, from the label and description for a menu entry.

Two new elements can be defined for a menu entry (not at the top level) in AssetCenter Web 4.2.

Element	Description
Label (or Label_ids)	this element is required for every menu entry (aka option or answer). This is the text that is displayed next to the icon.
Description (or Description_ids)	this element is used to display a long description in a tool tip for the icon.

Title and **Instructions** are now only used for the form title and instructions.

Clickable trace widget

The old tree menu had a read-only trace, presenting selections the user made before getting to the current page.

In AssetCenter Web 4.4.0, the trace widget is clickable and allows to go back directly to a **parent** page.

As a result, a new node has been added to be able to go to the root directly. The previous trace widget did not show the root.

Element	Description
Link text	Where does the text that shows for a link to a page in the trace widget come from? By default, the text comes from the menu option's **Label** that led to that page. This, of course, does not work for the first page the user was presented with, since no menu option led to that page. For the first page, the link defaults to **Start**.
	If you want a different text to lead to a page in the trace widget, you can set up the page with the TraceLabel element in the xml configuration file (or TraceLabel_ids). This gives you an opportunity, for example, to change the link label to the first page. Just set the TraceLabel to the text you want in the WizardMenu element at the root of your configuration file.
Other available options	
bClickableTrace	Option to control whether the trace is clickable. This option is true by default.
bShowTrace	Ability to show or hide the trace on a given page. By default, this is false at the root level, true at the other levels.

View items as icons or list

There are now two possible ways to display the available options. They can display as icons, the way it was in 4.1, or they can display in a list.

The new element ViewAnswersAs controls the type of display. It can be set to **list** or **icons**. The default is **icons**.

The list is a regular paginated list of 20 entries.

Elements controlling the list view can be set.

Element	Description
ListColumns	Semicolon separated list of option elements to display in a list. The default is Label.
ListColumnHeaders	Semicolon separated list of strings to display in the header. The number of headers taken into account will be the number of columns found in ListColumns. The headers can be defined as an externalized string using the syntax $$IDS(goupofmodules,stringid).

Done button

An optional **Done** button can be added to a page. It gives the user the ability to choose the current page as a selection, instead of having to choose among the menu entries.

When this button is clicked, the Id associated with the page, the currently gathered values (see Wizard behavior) are passed as parameter. The target elements for the page are used to determine where to send the selection to (TargetForm, TargetURL) and what parameters to pass (TargetAddNoParams, TargetParams), or the default target defined by the application.

The following elements control the new option.

Element	Description
bShowDoneButton	Controls whether to show the button or not. False by default.
DoneButtonLabel (or DoneButtonLabel_ids)	Label for the **Done** button. The default is **Done**. **Note:**

Wizard-like behavior

The tree menu was originally designed to return one value, the Id of the last selected node. In this case, all the information needed by the application is held into that final node, and can be retrieved using a simple call to `configurabletreemenu.getMenuEntry`.

This scenario does not work well when the values needed by the application are not associated with the final node, but on each of the selected node, and that knowing the last node is not enough to retrieve the previous selections. This is especially a problem when dealing with dynamic values, where you can't tailor a node knowing the path taken to get to it. (A typical example is

category/subcategory/product type/problem type, where problem type can give you the product type name, but not the subcategory or category name).

There was a way to get to all the selected nodes using the AncestorIds parameter passed with the final selection, and getting the previous nodes one by one. But in addition to be a lot of code to write, when used in conjunction with dynamic entries, this could have become database intensive.

Tailoring note: In AssetCenter Web 4.4.0, the tree menu provides a way for the application to specify which values it wants the tree to collect. Before calling configurabletreemenu.getSubMenu, the calling function sets the `_collectFields` element in the message. This is a comma-delimited list of element names. As the user selects a menu options, the tree menu code will search for elements on that node that match the names listed in the `_collectFields` parameter. It remembers the values for each field found. As the user clicks a new node, the code will add the new values to the previously found values.

Note that right now, this is set up only for Get-Services, where the category tree can collect Category, Subcategory, ProductType and ProblemType. In Get-Resources it is probably too dangerous to try and use it now (for catalog), as it could break the backwards compatibility of the old xml files.

Default values for dynamic entries

Dynamic menu entries can return any of the elements valid for a menu entry. Most of the time, though, the data source you get the data from does not provide all the values.

What can you do, then, to specify more elements that what is available in the data source?

In AssetCenter Web 4.1, you have several options:

■   For Icon and HasSubMenu you can specify a default value for all entries at the DynamicAnswers level. They will be applied to all the entries returned dynamically that don't have a value for the element (whether they are not

defined in the schema or there was no value for that field in the returned record).

- Write a postprocess script for the schema and set up all the default values in it.

- Use a script dynamic entry instead of a simple schema based dynamic entry and set up all the default values in it.

In AssetCenter Web 4.4.0, for the sake of simplicity, and to prevent the customers to write script as soon as they don't have enough data in their data source, the tree menu generalizes the declarative default values that existed for Icon and HasSubMenu. In the DefaultValues element, you can specify all the elements you want with the values you want (they still have to ultimately make sense for the tree menu or the application). These values can be constant, or can be calculated based on other fields dynamically returned for the entry, using the $$(fieldname) notation.

Inheritable page options

With AssetCenter Web 4.4.0, a lot more XML elements are available that control the way the page looks like:

Title	bShowTrace	DoneButtonLabel
Title_ids	bClickableTrace	ListColumns
Instructions	ViewAnswersAs	ListColumnHeaders
Instruction_ids	bShowDoneButton	ColumnCount

It is very likely that the customers will want to set these elements to the same value for all the pages presented by the tree. In order to avoid the tedious and error prone task to enter the same values on every nodes, these values are now inherited from the parent node. This means that if a value was set in an ancestor node, this value will apply to the current page unless the page overrides the value itself.

Tailoring note: If applications define additional page options, they have the ability to declare them inheritable. Before calling configurabletreemenu.getSubMenu, the calling function sets the `_inheritableElements` element in the message. This is a comma-delimited list of element names to inherit.

The Get-Resources line item category tree menu code uses this feature.

Impact on existing menu files

Changes in behavior

The menu files created for previous versions the will behave slightly differently in AssetCenter Web 4.4.0:

- The selection trace is now clickable: this is a new feature.

- The instructions are now inherited. On sub menu where no instructions were provided, which in version prior to AssetCenter Web 4.2 meant that no instructions were showing, the sub menu will now show the same instructions as the first ancestor providing instructions.

- The ColumnCount element is now inherited. It used to default to 2 when not specified. In AssetCenter Web 4.4.0, a page that does not specify a ColumnCount will use the first value found in the parent chain or 2 if none is found.

Validity of the menu definition files

The old menu definition files are invalid according to the syntax defined by the XML schema in 4.2. Here are the reasons:

1 The Label (or Label_ids) is now mandatory on the menu options. Title is not required any more.

2 The Id element must have a unique value in the menu file. This was the case before, since the code using the menu files counts on Id to be unique, but it was not enforced.

3 For the sake of clarity in the XML schema, the elements have been grouped by functional role: Id, AccessGroup, OptionGroup, TargetGroup, PageGroup. Since the XML schema definition cares about the order in which the elements are defined, some elements have moved that will make the file invalid according to the schema:

    a Access (belongs to the AccessGroup) must always be set just after Id, if Id is defined, or must be the first element of a page or option. In any case it must be set before Title.

    b ContextFilter (belongs to the AccessGroup) must be set after Access and Id, and before Title.

c Image (belongs to the OptionGroup) was set after Title and Instructions (PageGroup). It must now be defined before.

d The TargetGroup used to be placed in the middle of page definition elements (after Title and Instructions, before ColumnCount and Answers). Elements from this group must now be declared before the PageGroup, and in particular before Title and Instructions.

Note that we could very well build a conversion.

8 AssetCenter Web Administration

CHAPTER

This chapter includes instructions for administering your AssetCenter Web system.

Topics in this chapter include:

Accessing the Peregrine Portal Admin module

The Peregrine Portal administrator login page enables access to the Peregrine Portal Admin module. You use the Admin module to define the settings for your Peregrine system.

Note: After installing and building AssetCenter Web, you must log on as a ServiceCenter user with **getit.admin** rights to access the Admin module and administer the AssetCenter Web integration with ServiceCenter. For a list of access capability words and Adapter configuration instructions, see the section on AssetCenter Web security in this guide.

A default administrator, System, gives you access to the Admin module without being connected to a back-end system. After you configure your user name on the Common tab, you can also access the Admin module from the Navigation menu.

Important: When you change parameters using the Admin module, a `local.xml` file is created in the `\<appsrvr>\WEB-INF` directory (where `appsrvr` is the path to your application server) to store these parameters.

To access the Peregrine Portal administrator login page:

1   Verify that your application server (for example, Tomcat) is running.

2   In your Web browser Address field, type:

```
http://<hostname>:<port>/oaa/admin.jsp
```

3 Press Enter to open the Portal administrator login page.

4 In the Name field, type **System**.

No password is required on initial login.

5 Click **System Maintenance login.**

6 Click **Control Panel** to open the Control Panel page.

The activities available in the Admin module include:

Select this option	To do the following
Control Panel	View the status of connections to the back-end systems.
Deployed Versions	View the list of deployed applications with version numbers on this server.
Server Log	View activity on the AssetCenter Web server.
Settings	View and change settings for the Peregrine Portal.
Show Script Status	View and verify which application scripts are running. You can also start and stop scripts from this window.
Show Message Queues	View a list of all message queues.
Show Queue Status	View the current status of the queues: operational and unlocked, or suspended.
Adapter Transactions/Minute	View the transactions per minute for the back-end adapter.
IBM WebSphere Portal Integration	View the installed OAA portal components in the IBM WPS environment
local.xml File	Download the local.xml file

Using the Control Panel

Use the Control Panel page to check the status of the connections to the databases you are accessing with AssetCenter Web and your Web applications. You can also reset the connection between the Archway servlet and the adapters to the back-end systems.

To reset the connection between the Archway servlet and back-end system:

■   Click **Reset Peregrine Portal.**

A message at the top of the page indicates that the connections are reset.

Viewing the Deployed Versions

The Deployed Versions screen lists all of the packages that deploy during the installation, including the version number of each package.

To view the Deployed Versions list:

1   From the Activity menu, select **Deployed Versions**.

    A list of the installed packages opens.

2   Click **Print** for a printout of this list.

Using the Settings page

On the Activity menu, click **Settings** to open the current parameter settings. The Settings page is divided into tabs. The tabs that you see depend on the Web applications that you installed and the adapters that you use. The Common tab is available for all installations.

You configure the settings for the Portal, PortalDB,Web Application after the installation (refer to the AssetCenter Web Installation Guide). You can access the Settings page at any time to change the installation settings. Use the E-mail tab

to configure E-mail, so that users are notified by E-mail of their password when users have access to self-registration (see User self-registration on page 176).

To view Settings:

■ From the Activity menu, click **Settings**.

Each parameter on the tab has a description that guides you through the settings. The tabs you see on the Settings page depend on the Web applications you installed.

Setting parameters using the Admin module

When you make changes using the Admin Settings page, a `local.xml` file is created in the `C:\<appsrvr>\WEB-INF` directory. All changes to property settings are stored in this file. Restart the application server after making changes that are stored in `local.xml`.

To define a parameter:

1 Locate the setting you want to change and type the new parameter.

Note: If you have previously changed a setting and want to return to the default setting, click the **Click for default** link displayed in the description area for the parameter you want to revert. This link appears only when a setting is different from the default.

2 Scroll to the bottom of the page, and then click **Save**.

Note: You must click **Save** on each page before making changes to another setting.

3 From the Activity menu, click **Control Panel** > **Reset Peregrine Portal**.

An information message at the top of the Control Panel indicates that the server has been reset.

Choosing a login language

When you log in to the Peregrine Portal, you can choose from the Language pull-down list the language that the Portal displays. The default language is English, but you can enable additional languages.

Note: You can only enable additional languages if the language packs are
deployed.

To enable additional login languages:

1   Click **Settings** in the Control Panel.

2   Scroll down to the **Encoding**, **Locales**, and **Sessions** section.

3   In the Locales field type a comma-delimited list of the languages you want
to enable.

The first locale defines the default; in this case **en** for English, which already
appears in the field. A locale is specified by the ISO-639 language code,
which you can combine with the ISO-3166 Country code, separated with an
underline (_). For example, **fr** enables French; **en** and **en_US** specify U.S.
English, where dates are formatted Month/Day/Year; **en_GB** specifies British
English, where dates are formatted Day/Month/Year. The value
en_GB,fr,de,it specifies that British English, French, German, and Italian are
enabled.

4   Make sure that **Yes** is specified for **Enable Logout**. This is important because
you need to log out of the Peregrine Portal and log back in for your changes
to take effect.

Logging

You can use the Logging tab in the Admin Settings page to customize the logging of events in a server log file, whose default name is `archway.log`. A sample list appears in the text describing the Log domain text box.

The valid debug domains include the following:

acadapter	AssetCenter adapter (authentication, authorization, and adapter services)
scadapter	ServiceCenter adapter (authentication, authorization, and adapter services)
mailadapter	used for email
trigger	schema object trigger subsystem
bizdocadapter	BizDoc adapter (authentication, authorization, and adapter services)
presentation	how personalizations are delivered
personalization	wrench icon
weblication	personalization operations
archway	Archway services

ProcessorFactory	OAA internal request handling system (script, database and administration)
AdminController	Administrative request handling object
security	JAAS login modules to authenticate users
statistics	fundamental OAA statistics (moving averages)
oaaworkflow	workflow processes
templateengine	workflow templates
notificationservices	periodic script pollers that check for workflow assignments and workflow-related email notifications

The Log Level parameter allows you to specify the level of detail for the log information written to the log file. The All setting captures the most detail and the other settings specify various degrees or types of information collected for the specified Log domains. Possible values are: all, debug, info, warn, error, fatal and off in reverse order of detail. Typically, this setting should be left at warn or error so the logs indicate any significant problems encountered during production use. The more verbose settings of debug and info should be used during tailoring or problem isolation.

Logging format

You can specify in the Logging Format field the printing pattern of a log file. The logging format is composed of literal text and conversion specifiers. The details of the specifiers can be found in the following table, which can be found in its entirety, along with additional information, on the Apache.org web site at: *http://logging.apache.org/log4j/docs/api/org/apache/log4j/PatternLayout.html*

Logging format table

Conversion character	Effect
c	Used to output the category of the logging event. The category conversion specifier can be optionally followed by *precision specifier*, which is a decimal constant in brackets.
	If a precision specifier is given, then only the corresponding number of right-most components of the category name will be printed. By default the category name is printed in full.
	For example, for the category name "a.b.c" the pattern %c{2} is output as "b.c".
C	Used to output the fully qualified class name of the caller issuing the logging request. This conversion specifier can be optionally followed by *precision specifier*, which is a decimal constant in brackets.
	If a precision specifier is given, then only the corresponding number of right most components of the class name will be printed. By default the class name is output in fully qualified form.
	For example, for the class name "org.apache.xyz.SomeClass", the pattern %C{1} is output as "SomeClass".
	Note: Generating the caller class information is slow. Avoid it unless execution speed is not an issue.
d	Used to output the date of the logging event. The date conversion specifier may be followed by a *date format specifier*, which is enclosed in braces, such as
	%d{HH:mm:ss,SSS}
	or
	%d{dd MMM yyyy HH:mm:ss,SSS}
	If no date format specifier is given then ISO8601 format is assumed.
F	Used to output the file name where the logging request was issued.
	Note: Generating caller location information is extremely slow. Avoid it unless execution speed is not an issue
l [lower-case letter]	Used to output location information of the caller that generated the logging event.
	The location information depends on the JVM implementation, but usually consists of the fully qualified name of the calling method followed by the caller's source, the file name, and the line number enclosed in parentheses.
	Note: Though location information can be very useful, its generation is *extremely* slow. Avoid it unless execution speed is not an issue.

Conversion character	Effect
L	Used to output the line number from where the logging request was issued. **Note:** Generating caller location information is extremely slow. Avoid it unless execution speed is not an issue.
m	Used to output the application supplied message associated with the logging event.
M	Used to output the method name where the logging request was issued. **Note:** Generating caller location information is extremely slow. Avoid it unless execution speed is not an issue.
n	Outputs the platform-dependent line separator character(s), which offer practically the same performance as using non-portable line separator strings such as "\n" or "\r\n". Thus, it is the preferred way of specifying a line separator.
p	Used to output the priority of the logging event.
r	Used to output the number of milliseconds elapsed between the time when the application started and the time of the logging event.
t	Used to output the name of the thread that generated the logging event.
x	Used to output the NDC (nested diagnostic context) associated with the thread that generated the logging event.
X	Used to output the MDC (mapped diagnostic context) associated with the thread that generated the logging event. The X conversion character *must* be followed by the key for the map placed between braces, as in: `%X{clientNumber}` where `clientNumber` is the key. The value in the MDC corresponding to the key will be output.
%	The sequence %% outputs a single percent sign.

The format of the log file is determined by the Apache PatternLayout class.

Log file rollover

You can specify in the Log File Rollover Frequency Pattern field the frequency with which the log file is rolled over. The pattern that you enter is also used as an extension to name non-active files. By default the log file rolls over at midnight on the first day of each week, and logs a maximum one week's data. However, you can specify that the log files roll over at the following intervals: monthly, weekly, half-daily, daily, hourly, or every minute. Use the parameters in

the following table, which can be found in its entirety, along with additional information, on the Apache.org web site at *http://logging.apache.org/log4j/docs/api/org/apache/log4j/ DailyRollingFileAppender.html*

Date pattern	Rollover schedule
'.'yyyy-MM	The beginning of each month
'.'yyyy-ww	The first day of each week, depending on the locale
'.'yyyy-MM-dd	At midnight each day
'.'yyyy-MM-dd-a	At midnight and midday of each day
'.'yyyy-MM-dd-HH	At the top of every hour
'.'yyyy-MM-dd-HH-mm	At the beginning of every minute

The Apache DailyRollingFileAppender class determines the log file rollover frequency.

Viewing the Server Log

The Server Log provides a history of server events. The default file name is
`archway.log`.

To view the Server Log:

1 From the Activity menu, select **Server Log**.

A form opens with a drop-down list for you to select the log you want to
view.
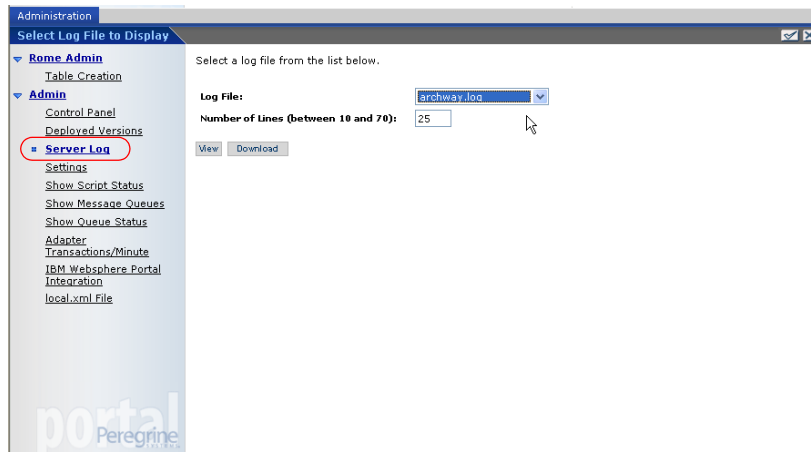
2 Click the drop-down and select the log file you want to view.

3 Set the number of lines to view.

4 Do one of the following:

- Click **View** to see the log file from your Web browser.
- Click **Download** to initiate the File Download wizard that downloads the
  `archway.log` file to a location of your choice.

Verifying Script Status

The Script Status page lists the name and status of any script that is currently running.

To verify the script status:

1   From the **Administration** Activity menu, click **Show Script Status** to display the **Status of Scripts** page that shows the name of each script.
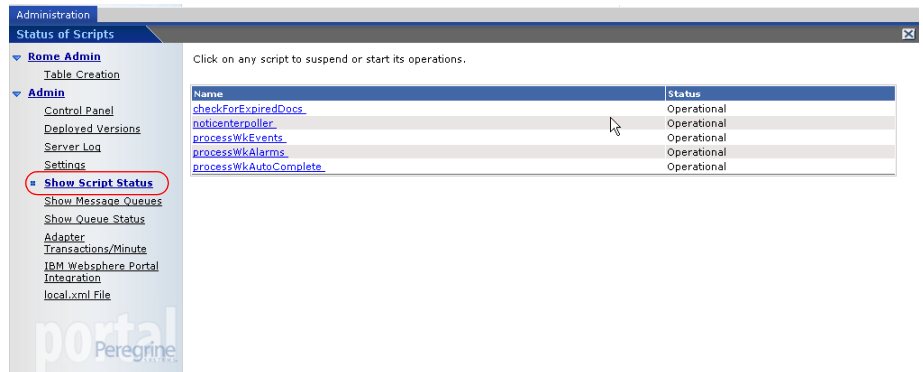
2   Click on the script to suspend it.

Displaying Message Queues

The Message Queues display whenever a queue has data waiting to be transferred.

To display message queues:

1   From the **Administration** Activity menu, click **Show Message Queues** to display the **Active Queues** page.
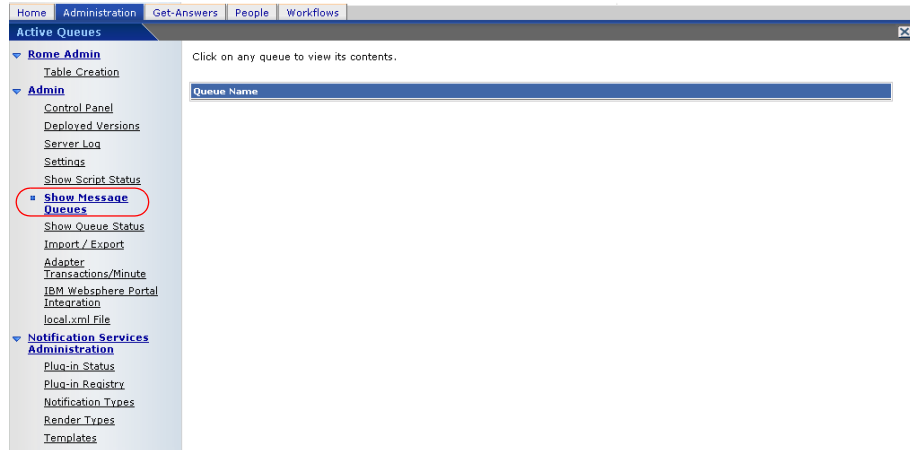
2   Click the queue name in the list to view the contents of a queue.

Showing Queue Status

Use the Show Queue Status option to verify or change the status of the message queues.

To show queue status:

1 From the Activity menu, click **Show Queue Status** to open the Queue Status page.
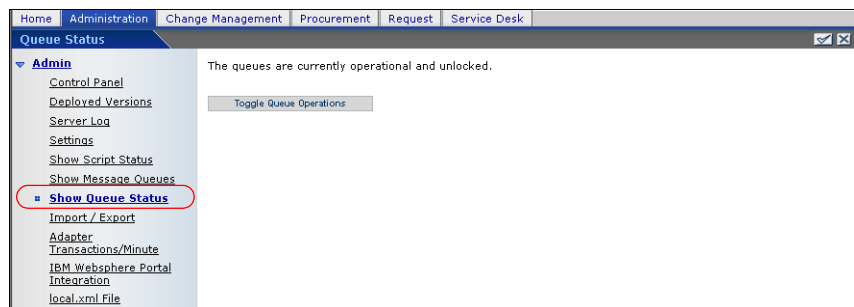
2 Click **Toggle Queue Operations** to change the status to **suspended**.

A warning message indicates that the Queue Status is suspended.
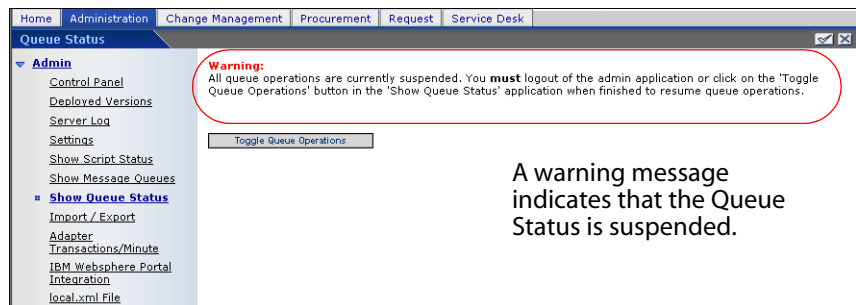
3 Click **Toggle Queue Operations** to return to the operational status.

Importing and exporting personalizations

You can move personalizations that you created in a development environment to a production environment. See the Personalization chapter in this guide for detailed instructions on importing and exporting the personalizations. Select the **Import / Export** option from the Admin activity menu to access the page.
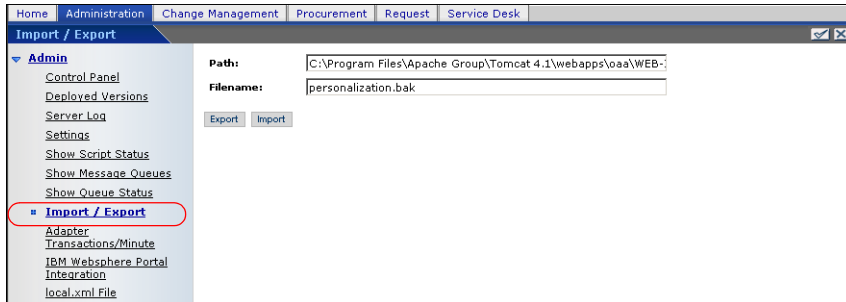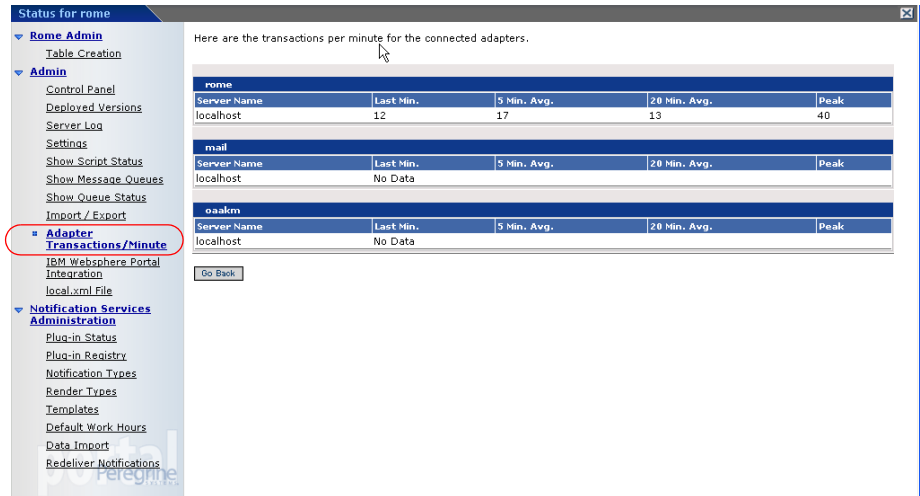
Viewing adapter transactions

You can track your adapter transactions by viewing the adapter Status page.

To view adapter transactions per minute:

■ From the Activity menu, click **Adapter Transactions/Minute** to open the
adapter **Status** page.

Using the IBM WebSphere Portal

You can generate an IBM WebSphere Portal Server web archive (war) file configured with references to installed OAA portal components.

To generate a war file:

1 From the Activity menu, click **IBM WebSphere Portal Integration** to open the **Portal Integration** page.

2 Enter the following information:

- source path
- destination path
- base URL

3 Click **Generate WAR File**.

Downloading the local.xml file

When you change parameters using the Admin module, a `local.xml` file is created in the `\<appsrvr>\WEB-INF` directory (where `appsrvr` is the path to your application server) to store these parameters. You can download the `local.xml`to preserve your settings if you want to test other settings and then replace the `local.xml` file your test created with your original `local.xml`.

To download the local.xml file:

1 From the Activity menu, click **local.xml File** to open the Download local.xml File page.

2 Click **Download**.

3 In the File Download dialog box, select **Open** or **Save**.

4 If you selected Save, specify the save location for the `local.xml` file.

Displaying form information

You can use the Admin module to configure Web application forms to display the location and file name of the current form.

To display form information:

1 From the Admin module, click **Settings** > **Logging**.

2 Scroll to the **Show form info** field, and click **Yes**.

3 Click **Save**.

4 For this particular setting, it is not necessary to Reset Peregrine Portal.

The name of the form is at the top of each form.

The form name is at the top of the page.

Displaying form details

You can also display detailed information about the current form. Click the **Display Form Info** button at the top right of the form. A separate window opens.

This is a partial example of the contents in the PortalDB tab.

View the contents in each tab for more detail about the form.

The form has the following tabs.

This tab	Contains
Script Input	the script that sends a request to the back-end system.
Script Output	the information returned by the script request to the back-end system.
User Session	details about the current user session, including browser type, back-end system version, and the access rights established for this user.
Log	a list of actions taken by the script to execute the form.
PreXSL	output from XSL before it gets rendered to the browser.
Browser Source	HTML source code for the current page.
BackChannel Source	HTML source code for frames where the data is stored.
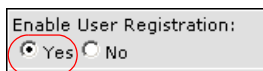
This tab	Contains
Application Channel Source	HTML source code for the shared applications.
Tab Source	HTML source code for tabs.
Menu Source	HTML source code for menus.
Sync/Update Window	HTML source code to synchronize with the page and reload.
Help	Help for debugging the window.

User self-registration

With the Admin module, administrators can choose to have end users self-register for new accounts from the login screen if the user is not already in the database. When the user registers, the system creates an Operator record and a Contact record for the new user with basic user login rights. See the Security chapter in this guide for more information about the registration process.

To enable users to self-register from the Login screen:

1   From the Admin module Settings page, click **Common**.

2   Scroll to **Enable User Registration**.

Click Yes to give users the ability to self-register for new accounts.

3   Click **Yes**.

Tip: When using an application with ServiceCenter as the back-end system, the first name and last name are reversed in the ServiceCenter contact record from the format used in an OAA Platform application.

ServiceCenter stores names in the format last name/first name. The OAA Platform stores names in the format first name/last name. As a temporary solution, you can change the way operator names are handled in ServiceCenter using the **Use Operator Full Name?** option in the

Environment records for Incident and Service Managements. Refer to the ServiceCenter Documentation for instructions.

Changing passwords

Using the Admin module, administrators can choose to have end users change their own passwords from the Home page.

To enable users to change passwords:

1   From the Admin module Settings page, click **Common**.

2   Scroll to **Enable Change Password**.

Enable Change Password:
Yes   No
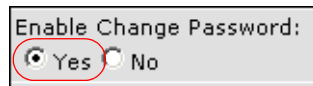
Click Yes to give users the ability to change their own passwords.

3   Click **Yes**.

Logging and monitoring user sessions

The `usage.log` file has a record of user logins that is in the `bin` directory of your application server installation. With this file, you can determine which application is in use and how many users access an application during a day.

Understanding the usage.log file

The following line shows an excerpt from a `usage.log` file.

```
127.0.0.1 - Tossi [04/Oct/2004:12:17:25 -0700] "GET portal
/portal/main/e_login_main_process.do HTTP/1.0" 200 0
```


Each login is on a line. Within one user session, each module logs only one line.

The following table shows the meaning of each element in the log entry.

Remote Host	Rfc931	User Login	Date	Request	Status	Bytes
127.0.0.1	-	Tossi	[04/Oct/ 2004:12:17 :25 -0700]	"GET portal/portal /main/e_login_main_ process.do HTTP/1.0"	200	0

This element	Contains
Remote Host	the remote host name or IP address if the DNS host name is not available or was not provided.
Rfc931	the remote login name of the user. This is always a dash because this information is not needed.
User Login	the user name authenticated to log in to the Peregrine Portal.
Date	the date and time of the request.
Request	the module accessed by the user. The name of the module is the first part of the GET parameter.
Status	the HTTP response code returned to the client. This value is always 200 to specify that it was a valid request.
Bytes	the number of bytes transferred. The number is always entered as 0, because this information is not needed.

9 | Back-end System Administration

The following sections describe administrative functions that must be performed outside of AssetCenter Web to support certain features available with AssetCenter:

- Configuring the Purchase Order Generation workflow on page 181
- Configuring the Product Catalog on page 182
- Installing and configuring the ACAdapter on UNIX on page 184

Configuring the Purchase Order Generation workflow

The Automatic PO Generation workflow in AssetCenter causes a purchase order to be created automatically each time a request is submitted in Get-Resources. You may want to disable this workflow in AssetCenter.

To disable the Automatic PO Generation workflow:

1   In AssetCenter, go to **Tools** > **Workflow** > **Workflow Schemes**.

2   In the list of Workflows, select **Automatic PO Generation**.

3   Select the **General** tab.

4 In the Validity section, End field, set the date to a time in the past.
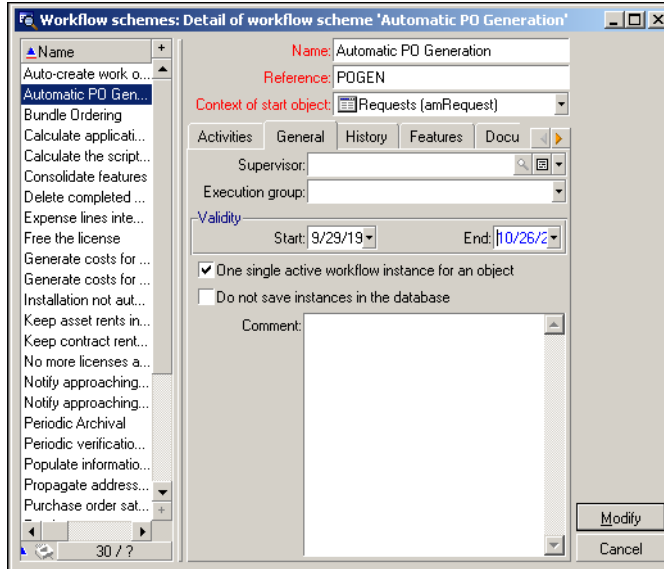
5 Click **Modify**.

6 Restart your application server.

Configuring the Product Catalog

This information applies only to AssetCenter 3.6.

AssetCenter Web uses the AssetCenter product catalog contained within the amProduct table. There are two areas in the catalog which require special configuration:

■ Certification field
■ Calculated field

Certification

AssetCenter Web uses the Certification field to determine the availability of items in the AssetCenter catalog.
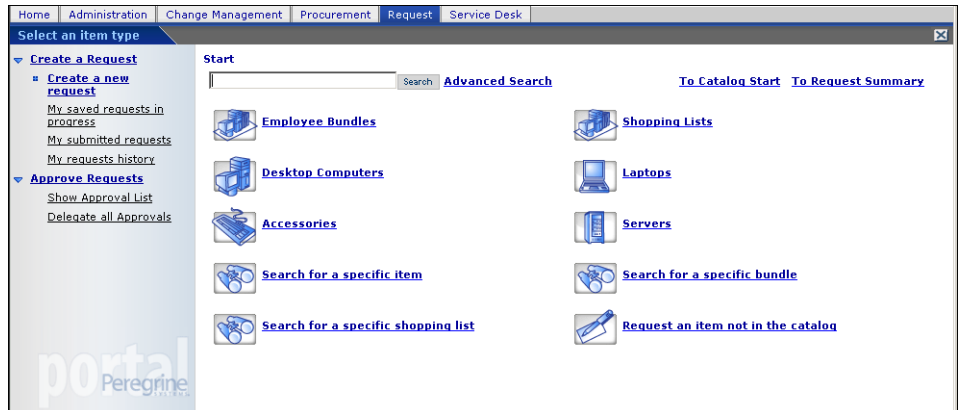
These buttons from the AssetCenter Web menu each drive a database call against the amProduct table. The queries executed are similar to the following for Desktop Computers:

```
SELECT lProdId,Brand,Model,mPrice FROM amProduct WHERE
(Certification LIKE 'Desktop%')
```

With the exception of the Bundle certification, all may be easily changed to meet your business requirements.

The Bundle certification is special within the Web application. Bundles are groups of items tied together for a specific purpose. For example, a Sales Laptop Bundle may consist of a laptop, PCMCIA NIC, Operating System software, and some applications. This relationship is built within the amProdCompo table, tying together several records from the amProduct table.

Calculated field: cf_Description

A calculated field is used as a descriptive name for records within the catalog. As identified in the Product schema, the field, Description, maps to the field cf_Description, a calculated field.

The following screen shows a sample of how AssetCenter's calculated fields may be used within AssetCenter Web to ease data presentation. Refer to your AssetCenter documentation for information about calculated fields.

Installing and configuring the ACAdapter on UNIX

If you are installing AssetCenter Web on UNIX and use AssetCenter as the back-end, you need the ACAdapter to interface with AssetCenter. Use the following procedures to install and configure the ACAdapter on UNIX.

Note: AssetCenter is a prerequisite for ACWeb.

To install and configure ACAdapter on UNIX:

1   Follow the AssetCenter documentation to install AssetCenter on the AIX, Linux, or Solaris system where OAA will be running.

Note: AssetCenter 4.4 on AIX requires the OpenSSL package from *http://www-1.ibm.com/servers/aix/products/aixos/linux/ download.html* Click the **AIX Toolbox Cryptographic Content** link.

2   Copy `amdb.ini` from a Windows AssetCenter installation that is configured to use the same AssetCenter database.

Note: Peregrine recommends that you copy it to the AssetCenter installation directory. You can copy the file to any location of your choosing where the application server user can read it.

Depending on your AC and Windows versions, this file may be in the `WINDOWS` or `WINNT` directory or your user's home directory under `Documents and Settings`.

3   Set the `LD_LIBRARY_PATH` environment variable (LIBPATH on AIX) for the user running your application server to include the `AssetCenter/bin`

directory and the directory containing the shared libraries for the database client that AssetCenter is using.

For example:

```
/usr/local/ac421/bin:/usr/local/oracle/product/8.1.6/lib
```

4   Set the `WPPCFGDIR` environment variable for the user running your application server to the directory where you copied `amdb.ini` in Step 2.

Important: Set the `WPPCFGDIR` variable to the directory where the file is, not to the file itself.

Note: For WebSphere Application Server 4.0.x on Linux and Solaris, you can set these environment variables on the General tab for the JVM running OAA.

For WebSphere Application Server 4.0.x on AIX, you must set the `LIBPATH` environment variable for the user running the WAS Admin Server before starting the Admin Server. You may want to edit a copy of `startupServer.sh` in `WebSphere/AppServer/bin` to set these variables.

5   Log in to `admin.jsp`, then click **Settings** and the **AssetCenter** tab.

6   Set the AssetCenter database name to match the database name defined in `amdb.ini`.

7   Set the AssetCenter library to match the version in your AssetCenter/bin directory.

For example, `"aamapi42"` for `libaamapi42.so`

Java will add the prefix and suffix back on when it looks for the file.

Optionally, set the AssetCenter library path to your `AssetCenter/bin` directory, including a trailing slash.

8   Click **Save**.

10 Security

CHAPTER

This chapter describes the different security configuration options available in AssetCenter Web. Topics in this chapter include, password and access rights for Health Insurance Portability and Accountability Act (HIPAA) support, default and custom security configurations, authentication, and alternate login pages.

By default, AssetCenter Web does not encode passwords sent over the network; however, passwords are stored in SHA-1 format (encrypted format) in the database. AssetCenter Web sends plain text passwords to the authenticating back-end databases and stores plain text passwords in a browser cookie if the user selects to **enable automatic login**. If you want to secure your AssetCenter Web passwords, you have three options:

- Enable Secure Sockets Layer (SSL) on your Web server
- Configure AssetCenter Web to use a directory service such as LDAP
- Enable your Web server to use Integrated Windows Authentication

In order to use SSL, you need to acquire a digital certificate. If your Web server has a certificate, then your AssetCenter Web login URL must include the `https` protocol indicator. After the user browser has made a secure connection to the Web server, all data transferred is encrypted. Refer to your Web server documentation for information on configuring SSL.

AssetCenter Web also supports authentication via a directory service such as LDAP. When you authenticate to a directory service, AssetCenter Web passes SHA hash encoding passwords to the service. For instructions configuring a directory service see .

AssetCenter Web also supports Integrated Windows Authentication. When this form of authentication is used, passwords are not actually exchanged between the browser and Web server, and the authentication process is kept secure. However, Integrated Windows Authentication is only supported by Internet Explorer browsers on Windows systems. For instructions configuring Integrated Windows Authentication see .

Back-end system security

This section includes information about how AssetCenter Web authenticates users and stores personalization changes in the ServiceCenter, AssetCenter, or Rome back-end system.

User account and password management

This section describes how the Administrator can manage user accounts and setup password formatting rules.

General administrative options

There are administrative options that apply to all the back-end adapters. These options are listed on the Common tab of the Administration page.

Enable Change Password:	Enables users to change their password
○ Yes ● No	from the Home module along with other profile information.
Allow current password to be new password:	When changing password, allow users to
○ Yes ● No	give their current password to be their new password.

Enable Change Password: The **Yes** option enables the display of the **Change Password** portal component for user accounts that are granted with the `getit.password` capability. The **No** option requires the administrator to change passwords for all users.

Allow current password to be new password: The **Yes** option gives administrators the ability to require users to enter a new password that is different from the current password when they are using the Change Password options.

Authentication with AssetCenter

When a user logs on to AssetCenter Web, the user name and password are validated against a corresponding operator record in the Employee table in AssetCenter.

When a user logs on, the AssetCenter back-end validates the user password, account status, and the password expiration according to the rules defined in each respective system. A generic error appears when a user fails to authenticate to any of the back-ends.

```
Sorry, your password has expired in at least one of the
supported targets. Please reset your password before you enter
the Peregrine Portal.
```

More back-end specific errors may be available in the `archway.log` file.

When one adapter returns an expired password code, the system redirects the user to the Change Password screen.

The user cannot log on to the system unless the user successfully resets the password.

Both the **Current Password** and **New Password** are sent to the back-end adapters. The AssetCenter back-end is responsible for verifying the current password and making sure that the new password complies with the rules and format for the password.

Generic error messages are displayed when a user has failed to reset the password. These messages can be customized by modifying the properties in the portal language string file in order to specify password format restrictions if desired.

Specific error messages may be found in the `archway.log`.

ServiceCenter capability words and AssetCenter user right keywords

Following is a list of available capability words and user rights keywords for AssetCenter Web functionality that can be assigned to a record in ServiceCenter or rome or a profile in AssetCenter.

Access	Description
getit.admin	Provides access to the OAA Admin module.
getit.advancedrequester	Enables access to advanced request form features in Get-Resources, including the ability to split request lines and assign request line items to different end users. This is useful for requesters who typically request items for a group of people.
getit.approver	Enables access to approve requests.
getit.buyer	Enables user to create and change the status of purchase orders. Only available with AssetCenter Procurement.
getit.pcardmanager	Enables users to create new Pcards and manage the rights for the Pcards they created. Only available with AssetCenter Procurement.
getit.pcarduser	Gives the user access to Pcards on the request and purchase order screens. These fields are not displayed if this access is not granted. Only available with AssetCenter Procurement.
getit.personalization.admin	Can set personalization options and save personalization changes as the default layout.
getit.personalization.default	Can change the layout and add or remove fields from AssetCenter Web interface.
getit.personalization.limited	Can only personalize features that have been exposed by a user with greater personalization rights.
getit.receiver	Enables access to receive items associated with a request. Only available with AssetCenter Procurement.
getit.requester	Enables access to create requests in Get-Resources.
getit.reserve	Enables approver to reserve resources from existing stock. Must have getit.approver access to function.

Access	Description
getit.view.shipment	Gives the user access to the shipment tracking tab available on a submitted request detail. Only available with AssetCenter procurement
oaa.forbidden	Reserved capability word to prevent access to all OAA users (cannot be granted to any user).

Note: In out-of-box ServiceCenter 5.1, assign the following capability words to Hartke and MAX.MANAGER so that they can approve requests:

> **Note:** getit.requester
> getit.advancedrequester
> getit.approver

Refer to the Administering ServiceCenter online help for detailed instructions on assigning capability words to operator records, or the AssetCenter Administration Guide for detailed instruction on how to add user rights to profiles.

AssetCenter sample security data

Sample profiles

AssetCenter Web provides several pre-set sample profiles in AssetCenter which combine user rights to give access to various modules. You can create profiles in AssetCenter with any combination of rights for each user.

AssetCenter Web includes the following sample profiles.

Profile	Privileges
getit.admin	Requesting
	Approving
	Purchasing
	Receiving
	System Administration
	PCard Administration
getit.default	Requesting
getit.full **Note:** AssetCenter Procurement only	Requesting
	Approving
	Purchasing
	Receiving
	PCard Administration

Profile	Privileges
getit.buyer	Requesting
Note: AssetCenter Procurement only	Purchasing
getit.requester	Requesting

When users self-register, they are initially assigned `getit.default` authority. You can update the AssetCenter employee records of those users who will need full or administration access.

Sample users

There are three sample user profiles provided within the AssetCenter demo database to illustrate various access rights for AssetCenter Web.

Full Name	User Name	Privileges
Michaela Tossi	Tossi	Requesting
Richard Hartke	Hartke	Requesting Approving Receiving
Michael Valentine	Valentine	Requesting Approving Receiving Purchasing Administration

From each user's profile tab in AssetCenter, you can also view the accessible tables for that user. Refer to the AssetCenter Administration Guide for detailed instructions on viewing user rights.

AssetCenter Web global access rights

Although initial login access for AssetCenter Web is validated against the user's corresponding profile in AssetCenter, global access rights can be granted for all users regardless of how their individual security is defined. For example, defining **getit.requester** AssetCenter Web even if it were not initially assigned to each user's profile in AssetCenter.

AssetCenter settings page of the Peregrine Portal Administration module.

To define global access rights within AssetCenter Web:

1 Open the Peregrine Portal Administration module in AssetCenter Web.

2 In the left menu pane, click **Settings**.

3 On the **Settings** page, click the AssetCenter tab.

4 In the AssetCenter settings page, update the appropriate field with the global access right(s) that you want to grant to all users in the following format:

```
<backend>(capability word)
```

where <backend> represents **ac** for AssetCenter.

Multiple default access rights may be granted by separating the capability parameter values with a semicolon (;). For example:

```
sc(getit.service;getit.itemployee)
```

Following is an example of how you update the appropriate settings page field for AssetCenter to grant all users the default right to create requests in AssetCenter Web.

Settings page	Field name	Sample field value
AssetCenter	Default Capability Words:	ac(getit.requester)

5 Scroll to the bottom of the form and click **Save**.

6 When the Control Panel page is displayed, click **Reset Peregrine Portal** to apply your configuration changes.

User registration

All AssetCenter Web users need a login account. In AssetCenter, the appropriate user rights are defined in the user's Profile. The user login is automatically authenticated in the back-end system.

If a user is attempting to log in for the first timewithout back-end authentication, the user is prompted for certain default information as shown in the following figure. The first four fields are required, as indicated by the arrows to the right of each field.

When the user clicks **Register**, the information is stored in the appropriate database. In AssetCenter, AssetCenter Web transforms this data into a Profile record that then passes to your AssetCenter system. An amEmplDept record is created with the user-supplied data and the default Profile `getit.default` is assigned.

Note: The appropriate back-end system adapter must be defined before the capability words are recognized. For example, if no adapter is defined for ServiceCenter, the ServiceCenter capability words are not used.

Basic registration information and login scripts are stored in the `.../oaa/apps/common/jscript/` directory. Login scripts are in the `login.js` file. If you want to make changes to the registration process, such as changing the way a user's password is defined, you can change the scripts in this directory or change the HIPPA security settings in the Rome database.

When a user account is created, the back-ends automatically populate the fields required by the account and password management. For example, the Rome back-end automatically calculates the Password Expiration Date.

Enabling the E-mail adapter

If users have the ability to self-register, you must make sure that the E-mail tab from the AssetCenter Web Admin module Settings page contains the MailAdapter name.

The MailAdapter is an implementation of JavaMail API 1.2 and supports the following mail protocols:

- POP3 for inbound mail
- IMAP for inbound mail
- SMTP for outbound mail

The MailAdapter also supports MIME type attachments in outbound e-mail.

Set the following parameters, as needed, on the E-mail tab of the Admin module Settings page.

Type the name of your MailAdapter in the Adapter field.

Troubleshooting the MailAdapter connection

You can check the status of the MailAdapter connection on the Control Panel. If the adapter shows as *disconnected*, check that the settings on the E-mail tab of the Settings page are correct. If you are still unable to connect, contact your system administrator for verification of the parameter values.

Authenticating users

You can configure the Peregrine OAA Platform to use one of five security authentication options:

■   Use the default configuration to authenticate users against Peregrine adapters. See Default security configuration on page 196.

■   Use a custom configuration to authenticate users against user-defined adapters such as LDAP or JDBC compliant databases. See Custom JAAS configuration on page 197.

■   Use a standard JAAS configuration to authenticate users against the Sun Microsystem's standard Java Authentication and Authorization Service (JAAS). See Standard Sun Microsystems JAAS configuration on page 206.

■   Use Integrated Windows authentication to authenticate users and pass the information to the Web application. See Integrated Windows Authentication on page 207.

■   Use an alternate login page and authenticate users against any of the other login options. See Creating an alternate login page on page 222.

Once a user is authenticated, the modules to which the user has access are defined by the back-end system. If you are using ServiceCenter for the back-end system, the user must have the appropriate capability words set in the Operator record in ServiceCenter in order to see the corresponding module in the web application.

Default security configuration

The default configuration authenticates users against a set of pre-configured JAAS login modules. By default, one JAAS login module is configured for each

registered Peregrine adapter. For example, if you are using both AssetCenter and ServiceCenter, then AssetCenter Web creates login modules for *both* the ACAdapter and the SCAdapter.

These login modules are *only* used to authenticate users. User access rights are derived from user profile records in the back-end systems (for example, ServiceCenter or AssetCenter). User access rights determine which modules the user can access and what tasks they can perform within those modules. For example, one user can open tickets only, while another has rights to approve tickets as well.

You do not have to do any additional configuration to use the default security configuration. AssetCenter Web automatically generates login modules for each Peregrine adapter installed on the system.

The default login module settings are:

Default Setting
`loginModule=com.peregrine.oaa.security.OAALoginModule`
`control flag=OPTIONAL`
`options=<none>`

Custom JAAS configuration

A custom JAAS configuration authenticates users against a set of JAAS LoginModules you define in a `local.xml` file. This file contains the settings to

use for each JAAS LoginModule. A `<jaas_config>` entry in `local.xml` has the following format.

```
<jaas_config>

   <jaasConfiguration>CustomConfig</jaasConfiguration>
   <CustomConfig>adapter1;adapter2</CustomConfig>

   <adapter1>
      <loginModule>Java class of login module</loginModule>
      <controlFlag>authentication behavior</controlFlag>
      <options>semicolon separated list of options</options>
   </adapter1>

   <adapter2>
      <loginModule>Java class of login module</loginModule>
      <controlFlag>authentication behavior</controlFlag>
      <options>semicolon separated list of options</options>
   </adapter2>

</jaas_config>
```

The following table describes how to use the XML tags and assign appropriate values.

Important: XML is case sensitive.

Use these XML tags	To do this
`<jaas_config>` `</jaas_config>`	Define a custom JAAS configuration. All JAAS configuration settings must be between these two tags.
`<jaasConfiguration>` `</jaasConfiguration>`	Define the name of your custom JAAS LoginModule. The value of this tag determines the tag name to use for the next tag. For example, if you create a custom configuration with the value `CustomConfig`, then you must use the tags `<CustomConfig>` and `</CustomConfig>` to define the list of adapters used.

Use these XML tags	To do this

`<CustomConfig>`
`</CustomConfig>`
This is a user definable tag.
	Define the list of *all* adapters that you want to use for authentication. Use semicolons between entries to specify multiple adapters.

	If the adapter name you list does not match a registered AdapterPool, then AssetCenter Web assumes the name is the logical name of a non-OAA LoginModule.
	AssetCenter Web attempts to authenticate users against each adapter you list. The values listed in this tag determine the tags names to use for each adapter. For example, if you create two adapters adapter1 and adapter2, then you must use the tags `<Adapter1>`, `</Adapter1>`, `<Adapter2>`, and `</Adapter2>` to define your adapters.

`<adapter1> </adapter1>`
`<adapter2> </adapter2>`
These are user definable tags.
	Define the JAAS LoginModule settings for each adapter. Each adapter *must* have both `<loginModule>` and `<controlFlag>` tags defined for it.

`<loginModule> </loginModule>`	Define the fully qualified class name of the JAAS LoginModule.
	This is *required* only when authenticating against non-OAA LoginModules (adapters). The default value is `com.peregrine.oaa.archway.security.OAALoginModule.`
	This is *optional* only when authenticating against Peregrine back-ends.
`<controlFlag> </controlFlag>` This tag is optional.	Define the authentication behavior of this LoginModule. The default value is REQUIRED.
	See JAAS LoginModule control flags on page 200 for a description of available options.
`<options> </options>`	Define the list of authentication options. Use semicolons between entries to specify multiple options. This is an *optional* setting for each JAAS LoginModule you use. See JAAS configuration options on page 201 for a description of available options.

JAAS LoginModule control flags

The following table lists the possible settings for the `<controlFlag>` tag. A JAAS LoginModule can have one of four behaviors:

Control flag	Authentication behavior
REQUIRED	If the user cannot be authenticated against the adapter, the login fails. Whether it succeeds or fails, authentication continues to the next LoginModule in the list.
REQUISITE	If the user cannot be authenticated against the adapter, the login fails. If it succeeds, authentication continues to the next LoginModule in the list.
SUFFICIENT	Authentication can proceed even if this LoginModule fails. If it succeeds, authentication does not continue to the next LoginModule in the list. If it fails, authentication continues to the next LoginModule in the list.
OPTIONAL	Authentication can proceed even if this LoginModule fails. Whether it succeeds or fails, authentication continues to the next LoginModule in the list. This is the default behavior.

Note: The controlFlag settings are case insensitive.

The overall authentication succeeds only if all Required and Requisite LoginModules succeed. If a Sufficient LoginModule is configured and succeeds, then only the Required and Requisite LoginModules prior to that Sufficient LoginModule need to have succeeded for the overall authentication to succeed. If no Required or Requisite LoginModules are configured for an application, then at least one Sufficient or Optional LoginModule must succeed.

By default, the controlFlag setting of all AssetCenter Web Web applications LoginModules is Optional. For most enterprises, this is the desired configuration.

The following table shows some sample scenarios and how the login process works.

Module Name	Status	Scenario 1	Scenario 2	Scenario 3
LoginModule1	required	pass	pass	fail
LoginModule2	sufficient	fail	fail	fail
LoginModule3	requisite	pass	pass	pass

Module Name	Status	Scenario 1	Scenario 2	Scenario 3
LoginModule4	optional	pass	fail	fail
Final Authentication		pass	pass	fail

In Scenario 1, authentication succeeds even though LoginModule2 fails. This is because the Required loginModule takes precedence over the sufficient loginModule.

In Scenario 2, authentication succeeds because the loginModules that failed are only Sufficient and Optional.

Scenario 3 authentication fails because a loginModule with a status of Required failed.

JAAS configuration options

The following tables list the possible settings for the `<options>` tag.

Standard JAAS Options

The following table lists the standard JAAS options available for all adapters.

Option	Use	Description
debug=true	optional	Instructs a LoginModule to output debugging information. The OAALoginModule logs debugging information to `stdout` and not to `archway.log`.
tryFirstPass=true	optional	The first LoginModule in the list saves the password entered and this password is used by subsequent LoginModules. If authentication fails, the LoginModules prompt for a new password and repeats the authentication process.
useFirstPass=true	optional	The first LoginModule in the list saves the password entered and this password is used by subsequent LoginModules. If authentication fails, LoginModules do not prompt for a new password.
storePass=true	optional	Stores the password for the user being authenticated.
clearPass=true	optional	Clears the password for the user being authenticated.

Peregrine JndiLoginModule options

The following table lists the options available to custom JAAS LoginModules using the Peregrine JndiLoginModule.

Note: The Peregrine JAAS LoginModule `com.peregrine.oaa.security.JndiLoginModule` is modeled after Sun's JndiLoginModule. The main difference is that an RFC 2307 (NIS over LDAP) compliant schema is not required. User must have "uid" and "userPassword" properties defined.

Option	Use	Description
user.provider.url	required	Use this option to provide the URL to the starting point in your directory service where you want to search for users.
		For example, `ldap://server/dc=peregrine,dc=com`
		Note: This option corresponds to the Java constant `Context.PROVIDER_URL`.
security.principal	optional	Use this option to specify which directory service user you want to use to authenticate non-anonymous queries of your directory service. Use the DN of the directory service user. For example, `uid=user,dc=peregrine,dc=com`
		Tip: To prevent user passwords from being visible to users, you should only set this option if you are using a directory server such as IPlanet where user passwords are SHA hashed by default.
		Note: This option corresponds to the Java constant `Context.SECURITY_PRINCIPAL`.
security.credentials	optional	Use this option to define the password for the `security.principal` user. This option should only be used in conjunction with the `security.principal` option.
		Note: If you are using a simple security authentication protocol, then this password may be passed as plain text.
		Tip: To safeguard this password, either enable SSL (set the `security.protocol=ssl` option) or use an `security.authentication` that protects passwords.
		Note: This option corresponds to the Java constant `Context.SECURITY_CREDENTIALS`.

Option	Use	Description
security.protocol	optional	Use this option to enable or disable an SSL connection between the JndiLoginModule and your directory server. This option has two possible values: `simple` (Default setting) `ssl` **Note:** This option corresponds to the Java constant `Context.SECURITY_PROTOCOL`
security.authentication	optional	Use this option to enable or disable anonymous binding to your directory service. Typically, this option has one of two values: **none** (Default setting) **simple** **Note:** If you do not specify a value for `security.principal` then `security.authentication` defaults to a value of none. Likewise, if you set `security.authentication` to `simple` but `security.credentials` is omitted or has zero length, then `security.authentication` resets to none. **Note:** This option corresponds to the Java constant `Context.SECURITY_AUTHENTICATION`.
user.search.scope	optional	Use this option to specify the number of levels to descend when searching for the user being authenticated by `user.provider.url`. This value must be an integer. The default value is 1. **Note:** This option corresponds to the Java constant `SearchControls.ONELEVEL_SCOPE`.
group.provider.url	optional	Use this option to provide the URL to the starting point in your directory service where you want to search for groups. For example, `ldap://server/dc=peregrine,dc=com` **Note:** This option corresponds to the Java constant `Context.PROVIDER_URL`.

Option	Use	Description
group.search.scope	optional	Use this option to specify the number of levels to descend when searching for a group. This option should only be used with `group.provider.url`. This value must be an integer. The default value is 1. **Note:** This option corresponds to the Java constant `SearchControls.ONELEVEL_SCOPE`.

group.search.objectClass	optional	Use this option to specify the name of the LDAP group objectClass. Valid values are:
		`groupOfNames` (Default value)
		`groupOfUniqueNames`
		`groupOfUrls`
		Note: Either `groupOfNames` or `groupOfUniqueNames` can be used to define static groups in LDAP, but they may not be used together.
		If you choose the `groupOfUrls` option, then you are configuring dynamic groups. No additional configuration settings are required to recognize dynamic groups.

storeIdentity=true	optional	Use this option to store a reference to the User being authenticated.
clearIdentity=true	optional	Use this option to clear a reference to the User being authenticated.

Example: Defining an LDAP custom configuration

The following XML code is an example of how to define a loginModule to authenticate users against an LDAP directory service.

Note: LDAP is not an adapter and does not imply any other functionality.

```
<settings>
   <jaas_config>
        <jaasConfiguration>myConfig</jaasConfiguration>
        <myConfig>ldap;ac;sc;rome</myConfig>
        <ldap>
          <loginModule>com.peregrine.oaa.security.JndiLoginModule</loginModule>
            <controlFlag>requisite</controlFlag>
             <options>
                setPreAuthenticated=true;
                user.provider.url=ldap://myldapserver:389/
                    ou=people,dc=mycompany,dc=com
            </options>
        </ldap>
   </jaas_config>
</settings>


Example with additional user capabilities stored in LDAP, see group.provider.url
option:

<settings>
   <jaas_config>
        <jaasConfiguration>myConfig</jaasConfiguration>
        <myConfig>ldap;ac;sc;rome</myConfig>
        <ldap>
            <loginModule>com.peregrine.oaa.security.JndiLoginModule</loginModule>
            <controlFlag>requisite</controlFlag>
             <options>
                setPreAuthenticated=true;
                user.provider.url=ldap://myldapserver:389/
                    ou=people,dc=mycompany,dc=com
                group.provider.url=ldap://myldapserver:389/
                    ou=groups,dc=mycompany,dc=com
            </options>
        </ldap>
   </jaas_config>
</settings>

Notes:
1) Character comparisons are case-sensitive. Be sure to match cases for all XML
tags and values.

2) Strip out all extraneous white space from with the values of elements. For
example, do not specify:
        <loginModule>
            com.peregrine.oaa.security.JndiLoginModule
        </loginModule>
Rather, use:
        <loginModule>com.peregrine.oaa.security.JndiLoginModule</loginModule>

White spaces are allowed only between semi-colon separated options within the
<options></options> tags.

Tip: To ensure your local.xml file contains valid XML formatting, open it using
InternetExplorer or some other XML viewing tool.
```

Standard Sun Microsystems JAAS configuration

The standard JAAS configuration option authenticates users against the Sun Microsytems formatted JAAS configuration. To enable the standard JAAS configuration, you must edit the local.xml file and add the following lines:

```
<jaas_config>
 <useStandardJAASConfiguration>true</useStandardJAASConfiguration>
</jaas_config>
```

If you choose to use the standard JAAS configuration, then you must also do one of the following two things:

- Specify the appropriate JAAS command line options when the container is started

    –or–

- Configure the `java.security` file in `$JAVA_HOME/jre/lib/security` for JAAS.

Command line options

The command line properties required for use of the standard file-based configuration are as follows:

```
java -classpath <list of jars> \
   -Djava.security.manager \
   -Djava.security.policy==java2.policy \
   -Djava.security.auth.policy==jaas.policy \
   -Djava.security.auth.login.config==jaas.config \
   <MyMainClass>
```

For `<list of jars>`, enter the list of jars used by your JAAS-enabled Java application.

For `<MyMainClass>`, enter the fully qualified class name of the Java main program class.

Integrated Windows Authentication

Windows Integrated Authentication (known as NT/Challenge Response in previous versions of Windows) is one of the ways Windows facilitates the authentication of users on a Web server. The process consists of a secure handshake between Internet Explorer (IE) and the Internet Information Server (IIS) Web server. The handshake lets the Web server know exactly who the user is, based on how they logged in to their workstation. This allows the Web server to restrict access to files or applications based on who the user is. Applications running on the Web server can use this information to identify users without requiring them to log in.

AssetCenter Web uses Integrated Windows Authentication as follows:

- The user logs in to a Windows XP/2000 workstation.
- The user starts the IE browser and navigates to the `login.asp` page.
- IE automatically sends user authentication information to IIS. The user's password is not transferred, but the Integrated Windows Authentication handshake between IE and IIS is enough for the server to recognize the user.
- The Web application login automatically detects the user by using the Integrated Windows Authentication/IIS server data.
- The user is logged in without requiring that a name and password be entered.

During this process, the back-end database authenticates and impersonates the Windows user with each of its adapters.

The following circumstance is an exception to the normal Integrated Windows Authentication login process:

- The Windows user name is not already registered in the back-end system. When this occurs, the Web application does not proceed with automatic login. This only occurs for users when the **Require Integrated Windows Authentication** option on the Admin page is set to No. The user sees another login screen and is asked for password verification. This step is an added security measure to prevent a user from accidentally logging in with administrative rights.

Setting up Integrated Windows Authentication

This section describes how to configure AssetCenter Web to use IIS for Integrated Windows Authentication while using Apache as the primary Web server. You can also follow these instructions if you use IIS as your primary Web server.

It is an eight-step process:

Step 1 Verify that all users have an Operator record in the appropriate back-end database. See Creating an Operator record on page 209.

Step 2 Install and configure AssetCenter Web with Apache and Tomcat. See Preparing to configure Integrated Windows Authentication on page 209.

Step 3 Set Web server properties for the `login.asp` file. See Setting Web server properties for the login.asp file on page 210.

Step 4 Set Web server properties for the `e_login_main_start.asp` file. See Setting Web server properties for the e_login_main_start.asp file on page 213.

Step 5 Set Web server properties for the `loginverify.asp` file. See Setting Web server properties for the loginverify.asp file on page 215.

Step 6 Set the **Require Integrated Windows Authentication** parameter, and optionally the **Default User Login Name** and **Default Login User Password** parameters from the AssetCenter Web administration page. See Setting the Admin parameters on page 216.

Step 7 Set the settings on the Common tab from the AssetCenter Web administration page. See Updating the Common tab URL settings on page 212.

Step 8 Optionally, define the **LogoutURL** from the AssetCenter Web administration page. This step is necessary when AssetCenter Web and IIS reside on different servers. See Setting up the LogoutURL on page 217.

The following procedures illustrate how to setup Integrated Windows Authentication using Windows 2000 as an example. If you are using Windows XP, the overall procedure is the same. The IIS Management Console is called Internet Information Services.

Creating an Operator record

All users must have a back-end database Operator record. Contact your Get-Answers, AssetCenter, or ServiceCenter administrator to verify that users have Operator records. Create an Operator record as needed.

Preparing to configure Integrated Windows Authentication

This section describes how to configure Integrated Windows Authentication if you use Tomcat as your application server, Apache as your Web server, and IIS for authentication.

1   Install and configure AssetCenter Web with Apache and Tomcat, and verify that you can log in through login.jsp.

2   On a server running IIS, create a virtual directory named oaa.

     This virtual directory must have read access and permission to run scripts.

3   From the AssetCenter Web deployment directory, copy the following files to the oaa virtual directory on the IIS server:

     login.asp

     loginverify.asp

     e_login_main_start.asp

     The default AssetCenter Web deployment directory is:
     C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa

Setting Web server properties for the login.asp file

Note: If you are using IIS for your Web server, go directly to Step 3.

1 On the IIS server, edit `login.asp` using a text editor.

Edit `<FORM… action…>` and change it from `login.jsp` to the absolute URL of `login.jsp` on the Apache server.

For example, change from:

```
<FORM name="f" action="login.jsp" method="post">
```
to:

```
<FORM name="f" action=
"http://<apacheserver.mycompany.com>/oaa/login.jsp"
method="post">
```

Note: If you are not using the default port (80), you must specify the port number on the URL.

2 Open the IIS Management Console (**Start**>**Programs**>**Administrative Tools**>**Internet Information Services**).

3 Click the `oaa` virtual directory.

4 Right-click `login.asp` and select **Properties**.

5 Select the **File Security** tab.

6 Click **Edit** in the **Anonymous Access and Authentication Control** section and set the permissions as follows:

    a Disable **Anonymous access**.

   **b** Require **Integrated Windows authentication**.
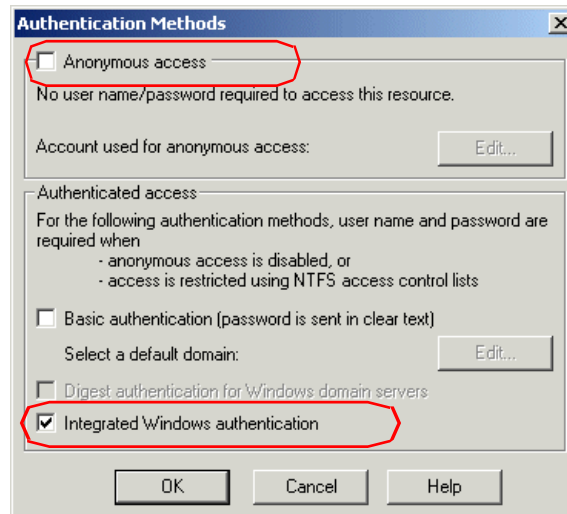
Clear the **Anonymous access** check box.

Select the **Integrated Windows authentication** check box.

7 Click **OK** on all windows displayed until you return to the Microsoft Management Console.

8 From Windows Explorer, update the following properties to `login.asp`.

    a Add the **Authenticated Users** group to the list of authorized users.

b Grant the following **Permissions** to the Authenticated Users group.

```
Read & Execute   Allow
Read             Allow
```
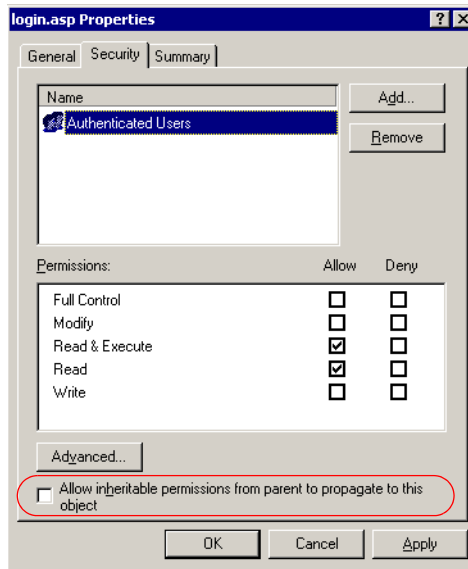

Make sure that only the Authenticated Users group is in this list.

Verify that the **Allow inheritable permissions from parent to propagate to this object** option is not checked.

c Clear the check box beside the **Allow inheritable permissions from parent to propagate to this object** option, then click **OK**.

Updating the Common tab URL settings

You need to set the Server URL and Login Verify URL parameters on the Common tab of the Admin Settings page.

To set the URL settings:

1 Log on the Peregrine Portal as a system administrator.

2 Click the **Administration** tab.

3   Click the **Settings** link.

4   On the Common tab, set the following parameters:

- Server URL - This must be the fully qualified Apache Web server / IIS server URL to the OAA virtual directory. The URL must include the port number if it is not 80.

- Login Verify URL - This must be the fully qualified IIS server URL to the OAA virtual directory. The URL must include the port number if it is not 80.

  Example: `http://DP8417:87/oaa_authentication`
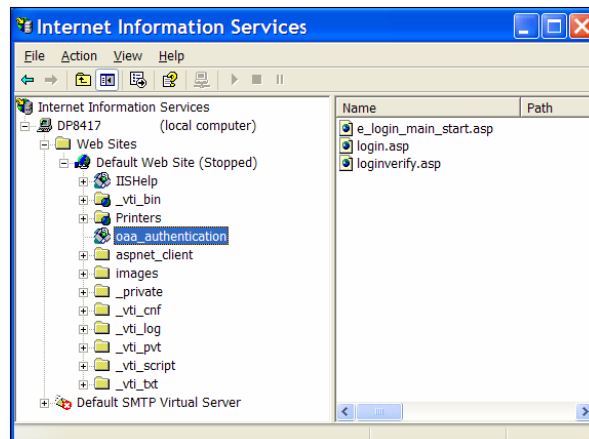
Setting Web server properties for the e_login_main_start.asp file

Note: If you are using IIS for your Web server, go directly to **step 3**.

1   On the IIS server, edit `e_login_main_start.asp` using a text editor.

Edit `<FORM... action...>` and change it from `e_login_main_start.do` to the absolute URL of `e_login_main_start.do` on the Apache server.

For example, change from:

```
<FORM name="f" action="e_login_main_start.do" method="post">
```
to:

```
<FORM name="f" action="http://<apacheserver.mycompany.com>
/oaa/e_login_main_start.do" method="post">
```

> **Note:** If you are not using the default port (80), you must specify the port number on the URL.

2 Open the IIS Management Console (click **Start** > **Programs** > **Administrative Tools** > **Internet Information Services**).

3 Click on the oaa virtual directory.

4 Right-click on e_login_main_start.asp and select **Properties**.

5 Select the **File Security** tab.

6 Click **Edit** in the **Anonymous Access and Authentication Control** section and set the permissions as follows:

    a Disable **Anonymous access**.

    b Require **Integrated Windows authentication**.
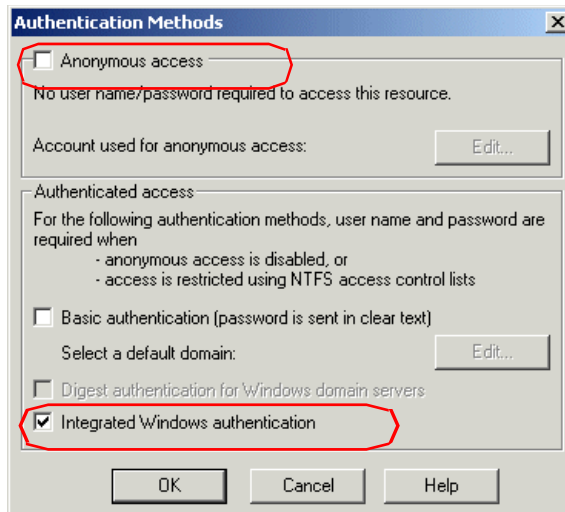
Clear the **Anonymous access** check box.

Select the **Integrated Windows authentication** check box.

7 Click **OK** on all the open dialog boxes until you return to the Microsoft Management Console.

8 From Windows Explorer, update the following properties to `e_login_main_start.asp`.

   a Add the **Authenticated Users** group to the list of authorized users.

   b Grant the following **Permissions** to the Authenticated Users group.

   ```
   Read & Execute   Allow
   Read             Allow
   ```
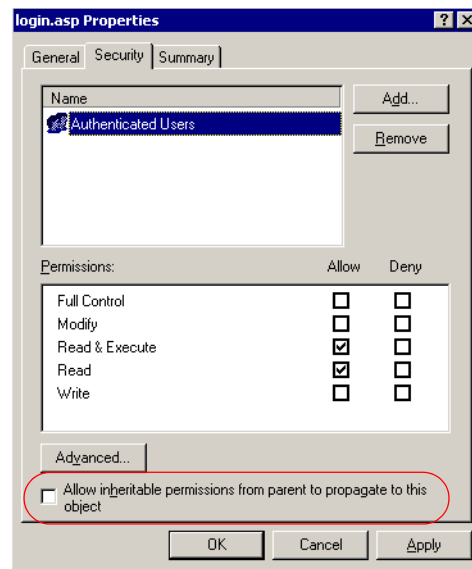

Make sure that only the Authenticated Users group is in this list.

Verify that the **Allow inheritable permissions from parent to propagate to this object** option is not checked.

   c Clear the check box beside the **Allow inheritable permissions from parent to propagate to this object** option, then click **OK**.

Setting Web server properties for the loginverify.asp file

1 Open the IIS Management Console (Start > Programs > Administrative Tools > Internet Information Services).

2 Click on the oaa virtual directory.

3 Right-click on `loginverify.asp` and select **Properties**.

4 Select the **File Security** tab.

5 Click **Edit** in the **Anonymous Access and Authentication Control** section.
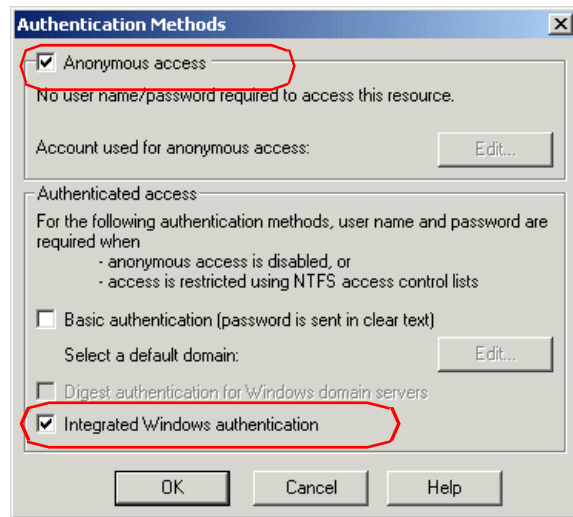
Select the Anonymous access check box.

Select the Integrated Windows authentication check box.

6 Verify that **Anonymous access** and **Integrated Windows authentication** have a check.

7 Click **OK** in all the open dialog boxes until you return to the Microsoft Management Console.

8 Close the Management Console.

Setting the Admin parameters

You must set the **Require Integrated Windows Authentication** parameter to Yes if you want only users who have a Windows account to log in. Users without Windows authentication can still have login capabilities by assigning a Default Login User Name.

Warning: The default login user has whatever capabilities you assign in the ServiceCenter or AssetCenter back-end. When you enable this feature, anyone can log in. Assign minimal user rights to this user.

To set Integrated Windows Authentication:

1 Open a Web browser.

2 Enter the following *URL: http://<webserver>/<oaa>/admin.jsp* in the browser address field (where `<webserver>` is the name of your Web server and `<oaa>` is the name of the virtual directory created during installation).

3 Login using the administrator name and password.

4 From the Administration Home page, click **Settings**.
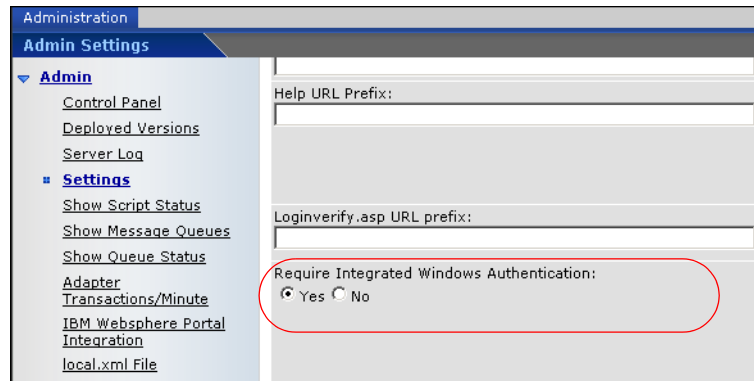
5 From the **Common** tab, set the **Require Integrated Windows Authentication** parameter to Yes.

6 To allow users without Windows authentication to login, assign a Default Login User Name, and optionally a password.

7 Click **Save**, then click **Reset Peregrine Portal**.

Setting up the LogoutURL

Note: This step is necessary when AssetCenter Web and IIS reside on different servers.

1 From the Administration home page (see Setting the Admin parameters on page 216), click **Settings**.

2 From the **Common** tab, set the **LogoutURL** setting to the URL you want users to go to if Integrated Windows Authentication fails or is not possible due to the user's current browser.

3 Click **Save**, then click **Reset Peregrine Portal**.

Testing the settings

Log in to your Peregrine Web application to make sure the access permissions are set correctly. The Integrated Windows Authentication settings are activated when you log in through a special login page named `login.asp`. Accessing your applications through the standard `login.jsp` page results in the users needing to log on as usual.

To test the settings:

1   Open a Web browser.

2   Enter the following *URL: http://<webserver>:<port>/<oaa>/login.asp* in the browser address field (where `<webserver>` is the name of your Web server, and `:<port>` is only required if it is other than port 80, and `<oaa>` is the name of the virtual directory created during installation).

3   Verify that access to AssetCenter Web is what you expected based on the settings you chose for the `login.asp` and `loginverify.asp` files.

Once you have verified this setting, all the users authenticated by Integrated Windows Authentication should now access Get-Answers with the `login.asp` URL.

Integrating with single sign-on tools

You can integrate AssetCenter Web with a single sign-on tool such as SiteMinder to eliminate displaying the AssetCenter Web login screen. When you integrate with a single sign-on tool, AssetCenter Web users browse to a special URL that obtains their user information from the sign-on tool and then automatically logs them in if the sign-on tool validates them. The following steps are for integrating AssetCenter Web with a third-party single sign-on tool. If you want to use Integrated Windows Authentication as your single sign-on tool, refer to Integrated Windows Authentication on page 207.

To integrate with a single sign-on tool:

1   Choose or create one user record for each single sign-on user you want to access AssetCenter Web. Each user record must have a password and a list of capability words or user rights.

Important: The back-end database user record is required to determine what portions of the AssetCenter Web interface the user can access.

2   Open a text editor such a NotePad.

3   Create a new JSP file to be the target of your automatic login URL.

You can use the following code as a template:

```
<%@ include file="jspheader.jsp" %>
<%

   // Add JSP code that obtains proper user name from
   // the third party single-sign on tool
   // ...

// Replace "user" with the user name obtained above
   String sUser = "user";

   // Turn on OAA pre-authentication
   user.setPreAuthenticated(true);
%>

<HTML>
<BODY>

   <FORM name="f" action="login.jsp" method="post">
      <INPUT type="hidden" name="loginuser" value="<%=sUser%>"
/>
   </FORM>

</BODY>
</HTML>

<SCRIPT LANGUAGE="JavaScript">
   self.document.forms[0].submit()
</SCRIPT>
```

4   Add any necessary JSP code to query your single sign-on tool for the name of the user who has been pre-authenticated.

Typically, these tools use HTTP headers to submit this information. See your single sign-on tool API documentation for details.

5   Save the file as `autologin.jsp` in your application server's presentation folder. For example:
```
C:\Program Files\Peregrine\Common\Tomcat4\webapps\oaa\
autologin.jsp
```

Note: The name you choose for the JSP file will be the file name required in the URL.

Testing access to AssetCenter Web from a single sign-on tool

You can use the following steps to test access to AssetCenter Web from your single sign-on tool.

To test your single sign-on settings:

1   Login to your single sign-on tool.

2   Open a browser and go to the following URL:

http://<server_name>/oaa/autologin.jsp

If you configured the login settings correctly you will be authenticated and redirected automatically to the AssetCenter Web home page.

Note: If you saved the automatic login page with a different file name, then use that file name instead of `autologin.jsp`.

AssetCenter authentication

AssetCenter Web can authenticate users using either NT or LDAP authentication. However, the two mechanisms are not entirely dependent.

Integrated Windows authentication with AssetCenter

If your AssetCenter user is not set up for integrated Windows authentication, you can still use integrated Windows authentication in AssetCenter Web, but you will need an employee for your user. (See Integrated Windows Authentication on page 207.) The employee's UserLogin is either: the NT user's name (the default); or the domain and user name in the format `<Domain>\<UserName>` if the `stripNtLoginDomain` entry in the `local.xml` file is set to False.

If your AssetCenter deployment is set up for integrated Windows authentication, AssetCenter Web cannot authenticate the user directly through AssetCenter. The AssetCenter Web user needs to be pre-authenticated by a trusted third party source. This source can be (and usually is) integrated Windows authentication. In any case, the user name of the third-party source must be the full NT name in this format: `<Domain>\<UserName>`. Further, the `stripNtLoginDomain` entry in the `local.xml` file must be set to False.

LDAP authentication with AssetCenter

AssetCenter Web can also authenticate users using LDAP. The mechanism is different from the way AssetCenter authenticates users with LDAP.

If your AssetCenter deployment is not set up for LDAP, you can still use LDAP authentication in AssetCenter Web. The login name for AssetCenter Web is the uid of a person; further, this uid *must* correspond to the UserName of a record in the amEmplDept (Employee) table in order to be able to perform the transactions in AssetCenter.

If your AssetCenter deployment is set up for LDAP, you need not use the LDAP authentication in AssetCenter Web because AssetCenter Web will use the AssetCenter LDAP authentication mechanism. The LDAP interface DLL, `nsldap32v50.dll`, which is delivered with AssetCenter, must be either in the startup directory of your Web Application Server (WebSphere, or Tomcat), or in your system wide path.

Creating an alternate login page

If you do not want to use the default Peregrine OAA login page, you can create your own login page that authenticates users and redirects them to the proper start page. Creating an alternate login page requires two basic steps:

Step 1 Create a login Web page with the necessary authentication parameters. See the following section, Creating a login Web page.

Step 2 Edit the `local.xml` file to specify the HTTP authentication method you want to use. See Specifying an alternate authentication method on page 223.

Creating a login Web page

Your custom login web page can be any HTML form that prompts for the following required parameters:

- Username
- Password

In addition, you can include optional login parameters such as:

- Display Language and Locale
- Time format
- Theme

A sample HTML login form, `login_sample.htm` is in the OAA deployment folder of your application server:

`<application server>\WEB-INF\oaa\`

Customize this sample HTML form using the following guidelines:

- Whatever custom login file you create becomes part of your login URL. For example, if you create a custom page called `my_login.htm`, then the login URL is *http://<server>:<port>/oaa/my_login.htm*
- You must specify the `basicauth` servlet in the form action. For example, `action="http://<server>:<port>/oaa/servlet/basicauth"`

- Users who fail to be successfully authenticated should see the page that is specified in the `_failURL` value. This can simply point to your login page so that the user can re-attempt login.

- The `basicauth` servlet does not encrypt usernames and passwords during login. You must enable HTTPS if you are concerned about password security on your intranet.

- There are no specific Administration page settings needed to set up a custom login page. You must define all login parameters in your custom login page.

- The following login parameters are available:

Login parameters	Description
loginuser	This is a required login parameter specifying the user name. You must specify a form input for this parameter.
loginpass	This is a required login parameter specifying the login password. You must specify a form input for this parameter.
_locale	This is an optional login parameter specifying the user's locale and regional display settings.
_timezone	This is an optional login parameter specifying the user's timezone.
_theme	This is an optional login parameter specifying which theme should be displayed in the Peregrine OAA Portal

Specifying an alternate authentication method

By default, Peregrine OAA uses HTTP basic authentication provided by the `HttpBasicAuthenticationManager` class. If you create a custom login page, you need to specify the alternate authentication method in the `local.xml` file.

To specify an alternate HTTP authentication method:

1 Stop your application server.

2 Using a text editor, open the `local.xml` file located at:

```
<application server>\webapps\oaa\WEB-INF\
```

3 Add the following entry to local.xml below the <settings> element (if the entry does not already exist):

```
<HTTPAuthClass>HttpAlternateAuthenticationManager</HTTPAuthClass>
```

4 Save the file.

5 Modify the `web.xml` file.

You will need to enable the AuthController servlet to establish a proxy for HTTP basic authentication.

a Using a text editor, open the `web.xml` file located at:

`<application server>\webapps\oaa\WEB-INF.`

b Add the following lines at the end of the last `<servlet>` definition.

```
<servlet>
    <servlet-name>AuthController</servlet-name>
    <display-name>AuthController</display-name>
    <description>A controller (decorator) servlet that can be used
to enable configurable auth protection of any
resource.</description>

    <servlet-class>com.peregrine.oaa.archway.AuthControllerServlet
    </servlet-class>
    <load-on-startup>2</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>AuthController</servlet-name>
    <url-pattern>/servlet/basicauth/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>AuthController</servlet-name>
    <url-pattern>/servlet/auth/*</url-pattern>
</servlet-mapping>
```

c Save the file.

6 Restart your application server.

Warning: Changing the HTTP authentication setting to the Alternate Authentication Manager exposes queries (including login names and passwords) in the URL. If you want to protect URL queries, then you must restrict access to this information through your Web server.

11 Troubleshooting

CHAPTER

This section offers solutions when trying to resolve administration problems.

The following problems can result from the Internet browser you use to view AssetCenter Web.

Issue	Solution
Navigation: When logged in to AssetCenter Web, using the browser Back, Forward, and Refresh buttons can cause unexpected behavior of AssetCenter Web forms.	Do not use the browser navigation or Refresh buttons with AssetCenter Web forms displayed.
When using the Microsoft Internet Explorer 5.5 browser, the following can occur: Icons fail to display in data set results. You cannot personalize Collections and Subdocuments. JavaScript errors appear during login (apparent only if the option to display JavaScript errors is turned on for the browser).	Upgrade to Internet Explorer 6.
After changing a theme using the Change Themes page, clicking the Go Back button does not return you to the Home page.	On the Activity menu in the sidebar, click My Home Page.
Using the Back button intermittently produces a page expired error message. This error most often appears when you attempt to return to a list screen from a detail screen.	Create a new search to regenerate your list. AssetCenter Web does not cache what is on the screen.

Index

PEREGRINE

U

W