

Peregrine

Get-Resources 4.1.2

Administration Guide

For Windows, AIX, Linux, and Solaris

© Copyright 2004 Peregrine Systems, Inc.

PLEASE READ THE FOLLOWING MESSAGE CAREFULLY BEFORE INSTALLING AND USING THIS PRODUCT. THIS PRODUCT IS COPYRIGHTED PROPRIETARY MATERIAL OF PEREGRINE SYSTEMS, INC. (“PEREGRINE”). YOU ACKNOWLEDGE AND AGREE THAT YOUR USE OF THIS PRODUCT IS SUBJECT TO THE SOFTWARE LICENSE AGREEMENT BETWEEN YOU AND PEREGRINE. BY INSTALLING OR USING THIS PRODUCT, YOU INDICATE ACCEPTANCE OF AND AGREE TO BE BOUND BY THE TERMS AND CONDITIONS OF THE SOFTWARE LICENSE AGREEMENT BETWEEN YOU AND PEREGRINE. ANY INSTALLATION, USE, REPRODUCTION OR MODIFICATION OF THIS PRODUCT IN VIOLATION OF THE TERMS OF THE SOFTWARE LICENSE AGREEMENT BETWEEN YOU AND PEREGRINE IS EXPRESSLY PROHIBITED.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This book, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems, AssetCenter, AssetCenter Web, BI Portal, Dashboard, Get-It, Get-Services, Get-Resources, Peregrine Mobile, and ServiceCenter are registered trademarks of Peregrine Systems, Inc. or its subsidiaries.

Microsoft, Windows, Windows NT, Windows 2000, SQL Server, and names of other Microsoft products referenced herein are trademarks or registered trademarks of Microsoft Corporation.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). This product also contains software developed by: Sun Microsystems, Inc., Netscape Communications Corporation, and InstallShield Software Corporation.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document. The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental. If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com. If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com. This edition of the document applies to version 4.1.2 of the licensed program.

Peregrine Systems, Inc.
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com



Contents

	About this Guide	9
	Book audience	10
	Related documentation	10
	Associated applications	10
	Terminology	11
	Typographical conventions	11
	Special elements	11
	Organization of the guide	12
	Need further assistance?	13
	Customer Support	13
	Documentation web site	13
	Education Services web site	14
Chapter 1	Architecture Overview	15
	Peregrine OAA Platform architecture	17
	OAA scalability	19
	Archway internal architecture	20
	Archway requests	21
	The Document Manager	23
Chapter 2	Get-Resources Overview	25
	Get-Resources features	26
	Requests	26
	Procurement	30
	AssetCenter workflows	33

	Bundle ordering workflow	33
	Routing request workflow	34
	Request approval workflow	34
	Automatic PO generation workflow.	35
	Request status workflow	36
Chapter 3	Customizing the Peregrine Portal	37
	Deploying the Classic theme variations	38
	Changing the default theme	39
	Changing the header graphic for all themes.	39
	Creating a custom theme	41
	Layers properties	45
	Changing framesets.	46
	Translating Get-Resources.	48
	Editing existing translation strings files	49
	Adding new translation strings files	49
Chapter 4	Using the Peregrine Portal	53
	Logging in to the Peregrine Portal.	54
	Using the Activity menu.	55
	Personalizing the Peregrine Portal	56
	Adding components	57
	Changing the layout	60
	Changing themes	62
	Displaying form information	63
Chapter 5	Using the Personalization Interface	65
	Personalization overview	66
	Forms and functions	66
	Personalization interface	67
	Adding and removing personalizations	68
	Configuring fields	70
	Configuring subdocuments	71
	Configuring collections	73
	Supporting personalization	74
	Activating Personalization.	75

Personalization tasks	77
Adding fields to a form	77
Configuring field attributes	78
Changing the label of a field	79
Changing a field to read-only	79
Making a field required	80
Changing the size and span of a field	80
Removing fields from a form	81
Making a schema visible to BVA portal components.	81
Moving Personalizations from a development to a production environment.	82
Chapter 6 Document Schema Definitions	85
Understanding Document Schema Definitions	86
Sample schema	87
How to use schemas	88
Schema extensions	89
When to use schema extensions	89
Creating schema extensions	90
Identifying the schema to extend	90
Locating the schema on the server	91
Creating the schema extension target folders and files	91
Editing the schema extension files.	93
Adding a new field to the Available Fields list.	93
Hiding an existing field from the Available Fields list	95
Changing the label a field displays in the Available Fields list	96
Changing the list of forms where a field is available or visible	97
Changing the physical mapping of a field	100
Changing the type of form component a field uses	101
Adding subdocuments to the Available Fields list	102
Schema Subclasses	106
Editing the schema subclass files	108
Editing the loadscript files.	109
Filtering a list of documents in a portal component	109
Filtering a list of documents in a field lookup.	110
Adding data validation for document updates or inserts	112

	Adding default values to a detail form	114
	Changing document data when a particular condition is met	116
	Schema Elements and Attributes	118
	<?xml>.	118
	<schema>	118
	<documents>	118
	<document>	120
	<attribute>	124
	<collection>	129
	Documents	131
	Subdocuments	132
Chapter 7	Modifying the Request Type and Item Type Selection Menus	139
	Configuring the hierarchical menu component	140
	General features of the menu component	140
	Syntax of a menu configuration file	141
	Configuring the request type selection menu	147
	Configuring the item type selection menu	149
Chapter 8	Get-Resources Administration	153
	Accessing the Peregrine Portal Admin module	154
	Using the Control Panel.	156
	Viewing the Deployed Versions.	157
	Using the Settings page	158
	Setting parameters using the Admin module	159
	Logging.	160
	Logging format	161
	Log file rollover	164
	Viewing the Server Log	165
	Verifying Script Status	166
	Displaying Message Queues	166
	Showing Queue Status	167
	Importing and exporting personalizations	168
	Viewing adapter transactions.	168
	Using the IBM Websphere Portal	169
	Displaying form information.	169

	Displaying form details	171
	User self-registration	172
	Changing passwords	173
	Logging and monitoring user sessions	173
	Understanding the usage.log file	173
Chapter 9	Back-end System Administration	175
	Configuring the Purchase Order Generation workflow	175
	Configuring the Product Catalog	177
	Certification	177
	Calculated Field: cf_Description	178
	Installing and Configuring the ACAdapter on UNIX.	178
Chapter 10	Security	181
	Password encoding methods	182
	Back-end system security	183
	Authentication with ServiceCenter or AssetCenter	183
	ServiceCenter capability words and AssetCenter user right keywords.	183
	AssetCenter sample security data	185
	ServiceCenter password security	186
	Get-Resources global access rights	187
	User registration	188
	Enabling the E-mail adapter	189
	Troubleshooting the MailAdapter connection	190
	Authenticating users	190
	Default security configuration	191
	Custom JAAS configuration	192
	JAAS LoginModule control flags	194
	JAAS configuration options	196
	Example: Defining an LDAP custom configuration	200
	Standard Sun Microsystems JAAS configuration	200
	Command line options	201
	Integrated Windows Authentication.	201
	Setting up Integrated Windows Authentication.	202
	Testing the settings.	210
	Integrating with single sign-on tools.	210

	Testing access to Get-Resources from a single sign-on tool	212
	Authentication models	213
	ServiceCenter authentication components	213
	OAA contact and operator associations	213
	Regular operator authentication	214
	Algorithm for looking up contacts	214
	Contact creation	215
	Contact-based authentication	215
	AssetCenter authentication	222
	Integrated Windows authentication with AssetCenter	222
	LDAP authentication with AssetCenter	223
	Creating an alternate login page	223
	Creating a login Web page.	223
	Specifying an alternate authentication method	225
Chapter 11	Troubleshooting	227
	Browser issues	227
	Navigation Issue	227
	Tomcat issues	228
	WebSphere Portal Server issues.	228
	Sorting issues	229
	Index.	231

About this Guide

Get-Resources, part of the Get-It suite of Employee Self-Service products from Peregrine Systems, provides a way for businesses to make the procurement of resources available to employees through the corporate intranet.

Get-Resources integrates with AssetCenter Procurement, AssetCenter 4.x 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.

This guide includes the following topics:

- Performing administrative tasks in Get-Resources
- Configuring Get-Resources for ServiceCenter or AssetCenter
- Understanding how users are identified in Get-Resources
- Using the Peregrine Portal
- Personalizing forms

Book audience

This guide is intended for administrators who configure and maintain Get-Resources. 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
- ServiceCenter or AssetCenter administration and functionality
- XML knowledge

Related documentation

Refer to the following documentation for additional information:

This manual...	Provides information on...
<i>Get-Resources Installation Guide</i>	installing and configuring the Peregrine OAA platform, Get-Resources, Java SDK, and application and Web servers.
<i>Get-Resources Release Notes</i>	any late-breaking documentation or known issues with Get-Resources. These are constantly updated and posted to the Customer Support web site. See <i>Need further assistance?</i> on page 13 for details on accessing the Customer Support website.

Associated applications

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

Note: You must install and configure ServiceCenter or AssetCenter before you can install and configure Get-Resources. Refer to the *Get-Resources Installation Guide* for instructions.

Terminology

The terminology used in this guide and in the Get-Resources interface is based on ServiceCenter 4.x and 5.x. and 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 .
<i>Italics</i>	Variables and values that you must provide appear in <i>italics</i> . New terms also appear in <i>italics</i> .
Monospace	Code or script examples, output, and system messages appear in a monospace font. <pre>var msgTicket = new Message("Problem"); ... msgTicket.set("_event", "epmc");</pre> <p>An ellipsis (...) is used to indicate that portions of a script have been 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.</p>
Sans Serif	Filenames, such as <code>login.asp</code> , appear in a sans serif 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

Element	Usage
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	<i>Chapter 1, Architecture Overview</i>
Get-Resources features and architecture overview	<i>Chapter 2, Get-Resources Overview</i>
Customizing the Get-Resources interface	<i>Chapter 3, Customizing the Peregrine Portal</i>
Configuring and using the Peregrine Portal	<i>Chapter 4, Using the Peregrine Portal</i>
Activating and using end user interface personalization	<i>Chapter 5, Using the Personalization Interface</i>
Using document schema definitions and schema extensions	<i>Chapter 6, Document Schema Definitions</i>
Customizing forms Note: This chapter is helpful if you are interested in changing the way the request type menu selection or the line item type menu selection looks, or if you want to add a purchase order type menu or a request line type selection menu.	<i>Chapter 7, Modifying the Request Type and Item Type Selection Menus</i>
Administering Get-Resources using the Admin module	<i>Chapter 8, Get-Resources Administration</i>
Configuring Get-Resources to work with ServiceCenter or AssetCenter	<i>Chapter 9, Back-end System Administration</i>
Security features	<i>Chapter 10, Security</i>

Need further assistance?

For help with this release, you can contact customer support, download documentation or schedule training.

Customer Support

For further information and assistance with ServiceCenter in general, 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 the information you need.

The KnowledgeBase contains informational articles about all categories of Peregrine products. 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 Get-Resources documentation, see the Documentation pages of the Peregrine Customer Support web site.

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 Get-Resources link.
- 5 Click a product version link to display a list of documents that are available for that version of Get-Resources.
- 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 via 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 Architecture Overview

CHAPTER

Peregrine Open Application Architecture (OAA) platform 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 Platform is the underlying architecture for many Peregrine products, including the Get-It suite of Employee Self-Service products which offers:

Get-It Product	Description
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.

Get-It Product	Description
Get-Resources™	Web-based solution that integrates with AssetCenter Procurement, AssetCenter 4.x 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 Administration, Service Desk, and Change Management (with ServiceCenter 5.0 and 5.1).

Peregrine OAA Platform 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 the Peregrine OAA Platform are monitored and maintained.

The base of Peregrine OAA Platform includes:

- 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—the Peregrine OAA Platform 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.

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

- 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.

- OAA Workflow (Get-Answers only)—enables workflow capabilities used by some Peregrine OAA Platform 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.

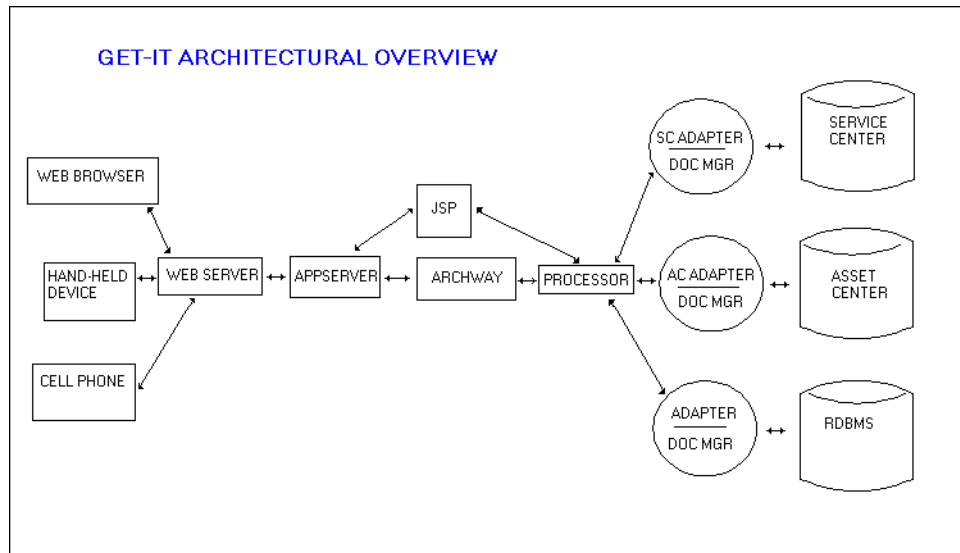
Peregrine OAA Platform architecture

Peregrine OAA Platform 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 Platform supplies Apache Tomcat for an application server with the installation. JRun, WebSphere, and WebLogic are also supported.
Common clients	Applications can be deployed via Web browsers (IE, Netscape), 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 could contact Archway and supply it with a new purchase request to be entered into the database.

All requests and responses are formatted using XML. For example, a problem ticket expressed in XML could appear as follows:

```

<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 Product News section 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 ServiceCenter.
- 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/oa/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 `/oa/servlet` mapping assumes that you are using the default URL mapping that Get-Resources automatically 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/oa/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 `/oa/servlet` mapping assumes that you are using the default URL mapping that Get-Resources 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/oa/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 `/oa/servlet` mapping assumes that you are using the default URL mapping that Get-Resources 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:

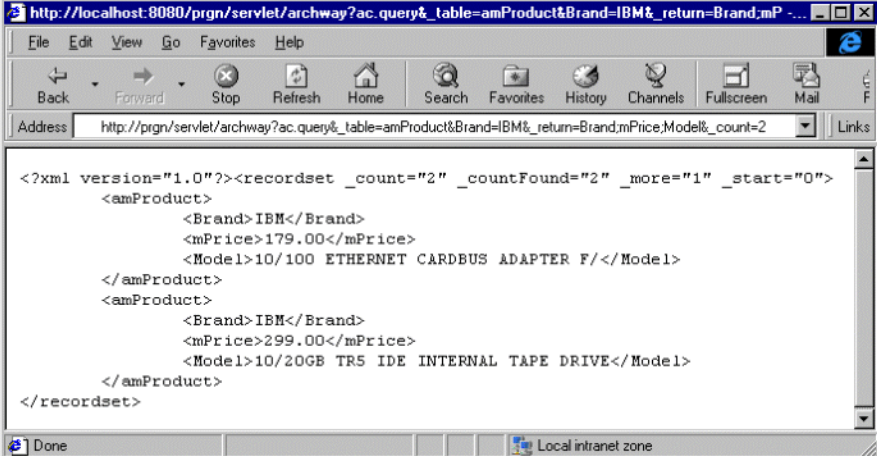
- `host name/oaaservlet/archway?sc.query&_table=probsummary&priority.code=1`

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

- `host name/oaaservlet/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.



```
<?xml version="1.0"?><recordset _count="2" _countFound="2" _more="1" _start="0">
  <amProduct>
    <Brand>IBM</Brand>
    <mPrice>179.00</mPrice>
    <Model>10/100 ETHERNET CARDBUS ADAPTER F</Model>
  </amProduct>
  <amProduct>
    <Brand>IBM</Brand>
    <mPrice>299.00</mPrice>
    <Model>10/20GB TR5 IDE INTERNAL TAPE DRIVE</Model>
  </amProduct>
</recordset>
```

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 Get-Resources Overview

CHAPTER

Get-Resources, part of the Get-It suite of Employee Self-Service (ESS) products from Peregrine Systems, enables employees to create requests for resources and services and streamlines the approval workflow of those requests throughout the organization.

Get-Resources integrates with AssetCenter or ServiceCenter to:

- Provide a web-based interface for employees to create and monitor the status of requests for resources and services.
- Streamline the request process by presenting employees with corporate approved products and services.
- Manage the approval workflow of requests throughout the organization.

Additional functionality is available with AssetCenter to:

- Fulfill requests based upon available stock prior to initiating a purchase.
- Create purchase orders for requested items that are not available in existing stock.
- Update and track the procurement process from the time that an order is placed to the time it is received.
- Enable employee acceptance or rejection of a request once it has been fulfilled.

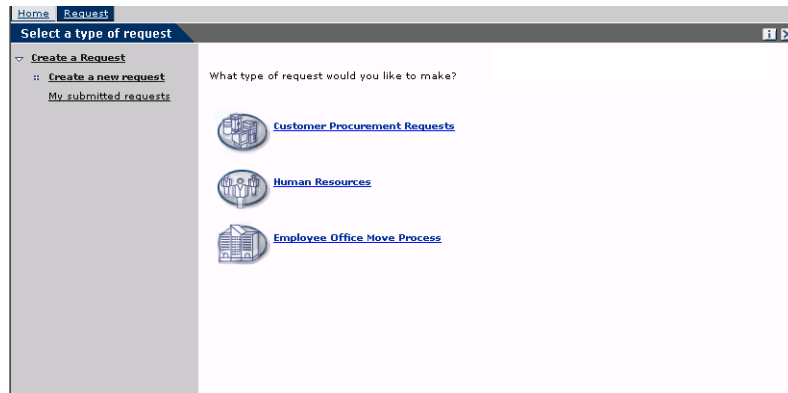
Get-Resources features

This section describes basic features that are available in Get-Resources.

Requests

The Request module in Get-Resources provides a web-based interface that enables users to create and view the status of requests for resources and services. Users with appropriate access rights can use this module to approve submitted requests.

The Request module displays a selection of request types. Users begin creating requests by selecting a request type, a feature which determines the options and screens that will be displayed for the rest of the creation process. The following is an example of the Request module **Select a type of request** form using ServiceCenter as the back-end.



See *Chapter 7, Modifying the Request Type and Item Type Selection Menus*, for more information on how to change the menu configuration.

Creating a new request

When a user selects an item from the catalog, the request detail screen is displayed for users to enter additional information for the request. Users can update values for the quantity or date the item is needed, as well as add financial information such as a cost center.

To Catalog Start

What is this for and when would you like it?

Purpose:

Date: Dec 27 2002

Time:

Who is it for?

EndUser: [Tossi](#)

First Name: Michaela

Phone:

What is the final destination?

Destination:

Address 1:

City:

Other information

Comment:

Attachments:

Quantity	Product/Description	Price
1	Compaq p500 Desktop computer w/ PIII500;124MB;13GIG	\$1,100.00

Grand Total: \$1,100.00

When using AssetCenter as a back-end system, Get-Resources provides additional functionality for adding purchasing card (PCard) information with the maximum limit on the card. It also has the ability to save the request for completion at a later date.

To Catalog Start

What is this for and when would you like it?

Purpose:

Date: Dec 27 2002

Time:

Who is it for?

EndUser: [Valentine](#)

First Name: Michael

Phone: (408) 422-5504

What is the final destination?

Destination: [002 - Office](#)

Address 1: 16 State Street

City: Burbank

Payment and approval information

Purchasing Card:

Max amount: US Dollar

Cost Center:

Project:

Signature Required:

Other information

Comment:

Attachments:

Quantity	Product/Description	Price
1	Compaq Compaq Compaq E500	\$2,049.00

Grand Total: \$2,049.00

My submitted requests

Once a request has been submitted, the user can view the status of that request by accessing [My submitted requests](#) from the left menu pane.

When using ServiceCenter as a back-end system, users can view pending approvals and access the approval log from the request status screen.

Quantity	Product/Description	Price
1	Compaq Desktop computer w/ PIII500;124MB;13GIG	\$1,100.00

When using AssetCenter as a back-end system, users can access the approval workflow and ship status from the request status screen.

Note: The ship status tab only shows for users that have the `getit.view.shipment` capability word.

Refer to *AssetCenter workflows* on page 33 for more information describing the graphical workflows available with AssetCenter.

Request Status		Approval Workflow	Ship Status
What is this for and when would you like it?		Payment and approval information	
Purpose:	testing	Purchasing Card:	
Date:	12/20/02	Max amount:	\$0.00
Time:	0:00	Cost Center:	IT
Who is it for?		Project:	
End User:	Valentine	Signature Required:	<input checked="" type="checkbox"/>
First Name:	Michael	Other information	
Phone:	(408) 422-5504	Comment:	
What is the final destination?		Attachments:	
Destination:	002 - Office	<input type="button" value="Add"/> <input type="button" value="Remove"/>	
Address 1:	16 State Street		
City:	Burbank		
Quantity	Product/Description	Price	
1	Palm device	\$0.00	
1	Workstation	\$0.00	
Grand Total:		\$0.00	
<input type="button" value="Return To List"/>			

Get-Resources also provides the ability to add the Status Review utility, a list of the user's active requests, to the Peregrine Portal.

Refer to *Personalizing the Peregrine Portal* on page 56 for detailed instructions on this function.

Approve Requests

Users with appropriate access can access a detailed view of requests submitted for approval from the Request module.

When using AssetCenter as a back-end system, users have the ability to update any details, perform a stock check to reserve an item from stock for the request before approval, or delegate approval for the request to another user.

Approval Required | **Approval Workflow** | [To Catalog Start](#)

What is this for and when would you like it?

Purpose: Backup laptop for testing

Date: Dec 24 2002

Time: 12:00 AM

Who is it for?

End User: Valentine

First Name: Michael

Phone: (408) 422-5504

What is the final destination?

Destination: 002 - Office

Address 1: 16 State Street

City: Burbank

Payment and approval information

Purchasing Card:

Max amount: US Dollar 0.00

Cost Center: IT

Project:

Signature Required:

Other information

Comment:

Attachments:

Quantity	Product/Description	Price
1	Compaq Compaq Compaq E500	\$2,049.00

Grand Total: \$2,049.00

Approver action and comments

Approver Comments:

Approval Action:

Add More Items | Submit | Discard Changes | Delegate This Approval

Users can also delegate approval for all requests from the main menu of the Request module.

Get-Resources provides the ability to add a list of requests that are waiting for approval. Refer to *Personalizing the Peregrine Portal* on page 56 for detailed instructions on this function.

Procurement

For users with appropriate access, the Procurement module provides the ability to create purchase orders for requested items and track the status of those purchase orders to the end of the receiving process. Refer to the *Get-Resources Configuration* chapter of the *Installation Guide* for information on accessing the Procurement module.

This feature is only available when using AssetCenter as a back-end system for Get-Resources.

Create a Purchase Order

When a request is approved, if there is no reservation for the requested item(s) from existing stock, Get-Resources displays the item(s) in a list when accessing the Procurement module. Users can select the item(s) from this list to create a purchase order.

Select an item to add to the cart

[Create a Purchase Order](#)
 :: [Create a new PO](#)
[My saved purchase orders in preparation](#)
[My submitted purchase orders](#)
[POs to review](#)

[Receiving](#)
[Search Purchase Orders](#)

Note: Using the more arrows, when displayed, will deselect currently selected items.

[Advanced Search](#) [To Request Summary](#)

Select	Quantity	Product/Description	Price	Action
<input type="checkbox"/>	1	Compaq Compaq Compaq E500	\$2,049.00	<input type="button" value="Add"/>

[To Request Summary](#)

The purchase order detail screen includes all of the information originally entered when the request was created. Users have the ability to update the purchase order with changes, save it for completion at a later time or submit it for the next procurement phase, the receiving process.

General		
Supplier:	Interleasing	Purpose: New employee
Shipping Information		
Contact:	Valentine, Michael	Location: /Burbank site/2nd Floor/002 - Office/
Delivery Information		
Date needed for:		Time:
Signature Required:	<input checked="" type="checkbox"/>	
Billing and Payment		
Contact:	Hartke, Richard	Location: /San Mateo site/Building 02/2nd F
Purchase Card Name:		Purchase Card Max Amount: \$0.00
What department is paying for this?		
Cost Center:	IT	
Complement of Information		
Attachments:	-none-	Comments:
Quantity	Product/Description	Price
1	Compaq Armada Notebook - E500	\$2,049.00
1	Compaq Compaq ultralight case (Nylon)	\$60.00

Users can access purchase orders that have been saved, submitted or automatically approved by a workflow within AssetCenter from the left menu pane of the Procurement module.

Receiving

Information for requested items that have been received are entered in the Receiving module of Get-Resources. Users can enter details such as quantity received, asset tags and serial numbers for items before submitting to the final phase of employee acceptance.

End-user acceptance

As items are received an end-user can access **My submitted requests** to review the item details for accuracy and accept or reject the item as necessary.

AssetCenter workflows

AssetCenter provides workflows to help you automate and formalize your business procedures. The following AssetCenter workflows are available for use with Get-Resources and are used by the system in this order:

- Bundle Ordering (AssetCenter 3.x only)
- Routing Request
- Request Approval
- Automatic PO Generation
- Request Status

Each of these workflows follows a default process established in AssetCenter for Get-Resources as shipped. You can modify the workflows to suit your business needs.

Note: It is important that you leave the first and last boxes in a workflow unchanged, since these boxes are linked to the workflows that precede and follow each workflow. Changing these boxes will break the links between the workflows and render them unworkable. Refer to your AssetCenter documentation for information on creating and modifying workflows.

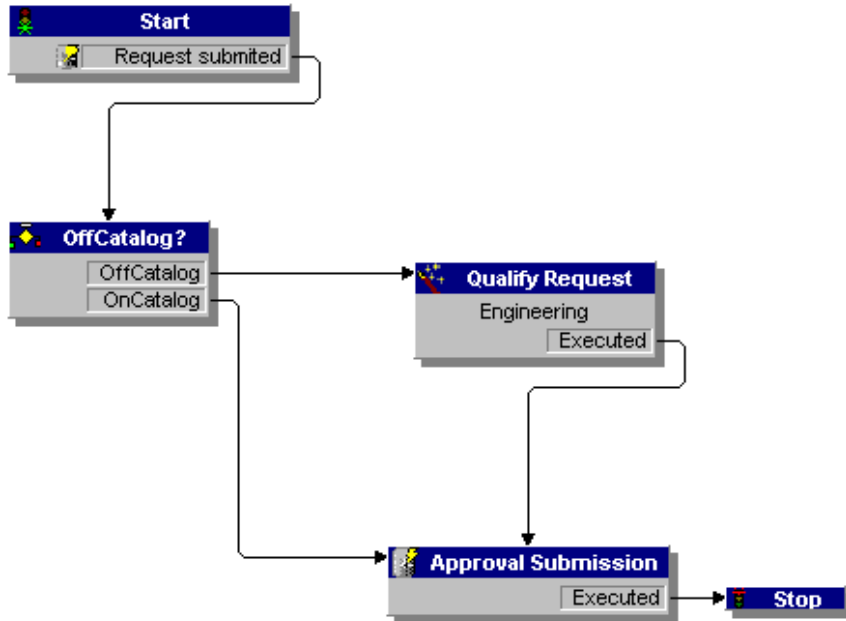
Bundle ordering workflow

When a request is submitted, it is checked by the Bundle Ordering workflow to see if a bundle request has been submitted. If true, the workflow then removes the bundle from the purchase order part of the request process. This is done so that the individual items that make up the bundle will be processed through to the purchase order, rather than the bundle itself.



Routing request workflow

The Routing Request workflow is activated when the status of a request is set to *submit*. By default, all requests are processed as OnCatalog request, so the Qualify Request box is not used. The last box in this workflow starts the approval cycle.

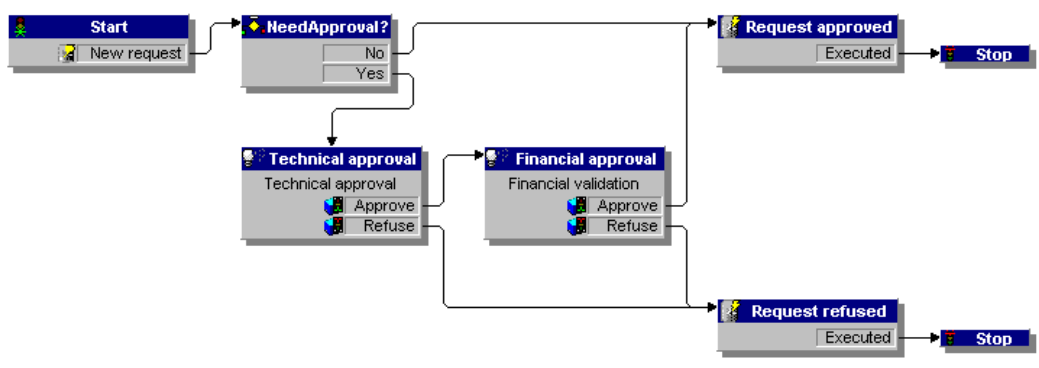


Request approval workflow

The Request Approval workflow shows the approval steps for a request that has been submitted. After the request has gone through the approval process, there are two possible results at the end of this workflow.

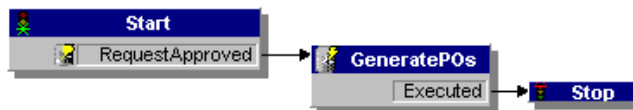
- If the request was approved, the approval status and the request status are set to *approved*. The request status determines what is displayed on the screen when the status of a request is viewed.
- If the request was not approved, the approval status and request status are set to *refused*.

If you create your own workflow in AssetCenter and you want it to show in Get-Resources, make sure that its Reference field starts with REQAPPR. Make sure that you have only one active workflow for a given request.



Automatic PO generation workflow

When the approval status of a request is changed to *approved*, the Automatic PO Generation workflow is activated. One purchase order per request is automatically created. You can change the process to create multiple POs by vendor or to have the system bundle several requests into one PO. If you do not want purchase orders to be generated automatically, refer to [Configuring the Purchase Order Generation workflow](#) on page 175 for instructions for disabling this workflow.



Request status workflow

After ordered items are received, the Request Status workflow changes the status of the request to *received*. When the status of a request is viewed, the status is now shown as *received*.



3 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 Portal 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*
- *Translating Get-Resources*

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 screenshots 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:

- *Accessible*, which makes the screen available to users who need high contrast colors or better accessibility support.
- *Baja*, which adds southwestern green and beige hues to the Classic design.
- *Quicksilver*, which adds silver and blue hues to the Classic design.
- *Sierra*, which 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 login to Get-Resources.

Changing the default theme

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

To change the default theme:

- 1 Open your Web browser and log in to the Admin module (`localhost/oa/admin.jsp`).
- 2 Click **Settings** > **Themes**. Change the following parameters:
 - a In the **Default skin/Theme** field, change the parameter to the name of the theme you want to use (for example, *Baja*).
 - b In the **Default stylesheet** field, change the parameter to the appropriate name for the CSS file (for example, *baja.css*).
 - c In the **Default XSL stylesheets** 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 When the Control Panel opens, click **Reset Server**.
- 5 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 Portal 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 41*.

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 46*.



- 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 Get-Resources administration page (admin.jsp).
- 4 Click **Settings > Themes**.
- 5 In the **Default Peregrine Portal logo** field, enter the name of your custom header logo.

Type your new image name.

Portal	Common	Service Desk	Portal DE	Themes	Web Application	Logins	ServiceCenter	XSL	E-mail
Internet Explorer stylesheet path:					Directory path for CSS stylesheets for Internet Explorer browser.				
css/									
Images path:					Set the images directory location. The directory name must be specified relative to the 'presentation' directory. Setting this allows you to move the default location of the images directory to another location. The default is "images/". You must add the slash at the end of this path.				
images/									
Skins/Themes:					Set the Skins directory location. The directory name must be specified relative to the 'presentation' directory. Setting this allows you to move the default location of the skins directory to another location. The default is "skins/". You must add the slash at the end of this path.				
skins/									
Default skin/Theme:					Set the Default Skin name for user sessions. Enter only the name of the skin. The default is "classic".				
classic									
Default stylesheet:					Set the CSS Stylesheet name for user sessions. To see all the styles used in The Peregrine Portal, click to see the Peregrine Portal Stylesheet Key . This file can be useful for customizing stylesheets. The default is "classic.css".				
classic.css									
Default XSL templates:					The default XSL template set to use when the user has not set a theme. This should be the same as the default skin when specifying a theme provided by Peregrine Portal.				
classic									
Default Peregrine Portal logo:					Set the global logo to be used in the application. The logo is skinned and is located at the root level of each skin directory in Themes. To add a new custom logo, add it to the skin template. Type in the name for the new logo image. Instructions for adding new images are included in the Peregrine Portal Tailoring Guide. The default logo is "getit_header_logo.gif".				
getit_header_logo.gif									
Application Tab Order:					List one module from each of the tab groups in the order that the tabs should appear. Tabs that are omitted will appear at the end of the list in no particular order.				
portal									

- 6 Scroll to the bottom of the page, and then click **Save**.
- 7 When the Control Panel opens, click **Reset Server**.
- 8 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 Get-Resources.

To create a custom theme:

- 1 Copy classic theme images, stylesheets, and XSL templates. These files are located at:
 - Images. `<application server>\oaa\images\skins\classic`
 - Stylesheets. `<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`
 - Stylesheets. `<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. Get-Resources 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
<code>.ActionButton</code>	The style used on buttons throughout the Portal.
<code>.ActiveMenuLink</code>	Used when the mouse hovers over a menu link.
<code>.ActiveModuleMenu</code>	Designates the currently-selected page within the navigational subset.
<code>.CurrentModuleMenu</code>	Designates the currently-selected navigational subset.
<code>.FormTitle</code>	Used for the title of forms. Normally used to title DocExplorer window content.

Style Name	Style Description
.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 stylesheet colors match your image colors.

Note: You can see a detailed stylesheet key in the themes Administration section of the Portal. To access the stylesheet key, locate the Default stylesheet field on the Themes tab of the Admin Settings page and click the **Peregrine Portal Stylesheet Key** link.

Portal	Common	Service Desk	Portal DB	Themes	Web Application	Logging	ServiceCenter	XSL	Email
Internet Explorer stylesheet path:					Directory path for CSS stylesheets for Internet Explorer browser.				
css/									
Images path:					Set the images directory location. The directory name must be specified relative to the 'presentation' directory. Setting this allows you to move the default location of the images directory to another location. The default is "images/". You must add the slash at the end of this path.				
images/									
Skins/Themes:					Set the Skins directory location. The directory name must be specified relative to the 'presentation' directory. Setting this allows you to move the default location of the skins directory to another location. The default is "skins/". You must add the slash at the end of this path.				
skins/									
Default skin/Theme:					Set the Default Skin name for user sessions. Enter only the name of the skin. The default is "classic".				
classic									
Default stylesheet:					Set the CSS Stylesheet name for user sessions. To see all the styles used in The Peregrine Portal, click to see the Peregrine Portal Stylesheet Key . This file can be useful for customizing stylesheets. The default is "classic.css".				
classic.css									
Default XSL templates:					The default XSL template set to use when the user has not set a theme. This should be the same as the default skin when specifying a theme provided by Peregrine Portal.				
classic									
Default Peregrine Portal logo:					Set the global logo to be used in the application. The logo is skinned and is located at the root level of each skin directory in Themes. To add a new custom logo, add it to the skin template. Type in the name for the new logo image. Instructions for adding new images are included in the Peregrine Portal Tailoring Guide. The default logo is "getit_header_logo.gif".				
getit_header_logo.gif									
Application Tab Order:					List one module from each of the tab groups in the order that the tabs should appear. Tabs that are omitted will appear at the end of the list in no particular order.				
portal									

- 5 Save your theme stylesheet 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 about editing layers see [Layers properties on page 45](#).
- 7 Open and edit any XSL stylesheets you want to change.

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

The XSL stylesheets determine how Get-Resources displays form components in the main portal frame.

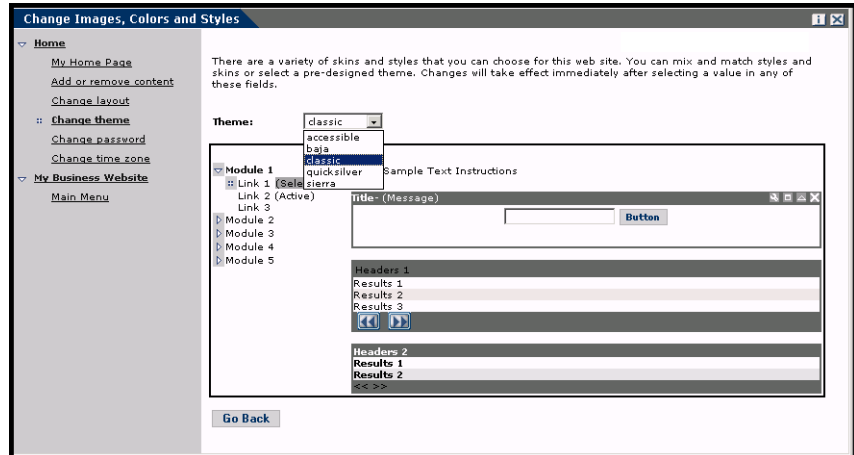
The following table lists the XSL stylesheets you can change.

To change	edit this XSL stylesheet
Attachment picker	attachments.xml
HTML form generation	basic-form.xml

To change	edit this XSL stylesheet
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;">

</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%">&nbsp;&nbsp;&nbsp;</td>
<td nowrap width="3" align="right" valign="top">
">
</td>
<td nowrap align="right" valign="top" width="100%" background="<%=
Archway.getSkinImagePath("backgrounds/rt_tile.gif", user ) %>">
">
</td>
<td nowrap><font class="userBarText" size="1" face="Arial, Helvetica,
sans-serif"><%=userTitle%></font>&nbsp;&nbsp;&nbsp;</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;">&nbsp;&nbsp;&nbsp;
</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>
      </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>
      </frameset>
    </frameset>
  </frameset>

```

Translating Get-Resources

Out-of-box, all Peregrine OAA web applications are provided in English. You can order translated versions of the core Peregrine OAA web applications by purchasing a language pack. Peregrine OAA 4.1.2 language packs are available for the following languages:

- French
- Italian
- German

Note: Not all Peregrine OAA web applications will 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 49*.

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 strings files on page 49*.

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 50*.

Editing existing translation strings files

You can make edits, additions, and deletions to string files outside of Peregrine Studio 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.

- 2 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"
```

- 3 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"
```

- 4 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 strings 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 *Translating Get-Resources* on page 48.

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.

Character Encoding	Character Set
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 Server** 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.

4 Using the Peregrine Portal

CHAPTER

The Peregrine Portal 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 Get-Resources uses. For more advanced changes to the portal, see the chapter on *Customizing the Peregrine Systems Portal*.

Topics in this chapter include:

- *Logging in to the Peregrine Portal* on page 54
- *Using the Activity menu* on page 55
- *Personalizing the Peregrine Portal* on page 56

Logging in to the Peregrine Portal

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

- A user login screen—`http://<server>/oaa/login.jsp`
- An administrator login screen—`http://<server>/oaa/admin.jsp`

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

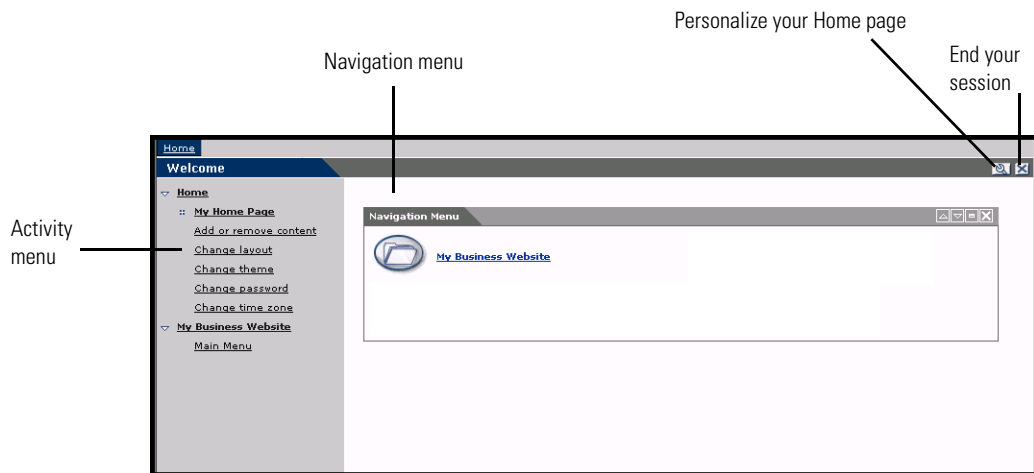
Note: This chapter discusses the features available with a user login. For more information about the administrator login, see the chapter on *Get-Resources Administration* in this guide.

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* on page 139.



The screenshot shows the Peregrine Portal login page. At the top left, the text "Peregrine Portal" is displayed in a large, blue, sans-serif font. To the right of the text are three small, square images: a close-up of an eye, a person standing in a doorway, and a person walking. In the top right corner, there is a logo that says "Powered by Peregrine" with a stylized bird icon. Below the header, there is a "Login" tab. The main content area has a "Welcome" message and a "Login" link. The login form includes a "User Name:" field with a search icon, a "Password:" field, and a "Language:" dropdown menu set to "English". A "Login" button is located below the form fields. The background of the page is light gray with a dark blue header bar.

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.



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 Portal

By default, the Navigation menu is displayed on the Peregrine Portal. You can personalize the Peregrine Portal to add Get-Resources 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:

Get-Resources Utilities

This component	Provides
Requests to Approve	List of requests that require your approval.
Status Review	Displays the list of your active requests so you may review their status.

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 5.0+ or Netscape 6.1+.

Administration components

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

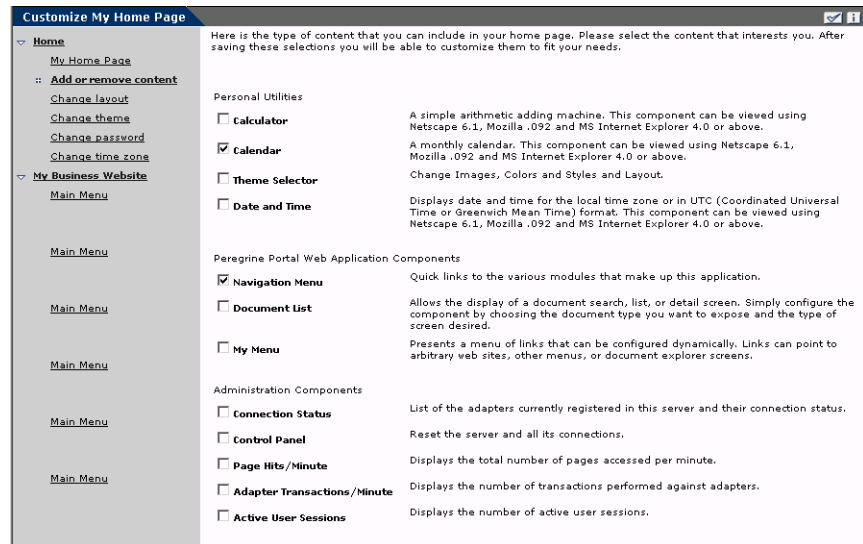
This component	Provides
Connection Status	A list of the adapters currently registered in this server and their connection status.
Control Panel	A button to reset the server and all its connections.
Page Hits / Minute	A list of the total number of pages accessed per minute.
Adapter Transactions / Minute	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 containing a list of the available components.



- 2 Select the components you want to add to 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 are displayed. The following example shows the Requests to Approve and Status Review utilities for Get-Resources.

Requests to Approve (3 requests)					
Action	Request Number	Request Total	Requester	Purpose	Approval Activity
	REQ001006	\$763.05	Hartke	asdf	Financial approval
	REQ001008	\$7,306.65	Hartke	asd	Technical approval
	REQ001020	\$60,461.39	Hartke	dfg	Technical approval

Submit

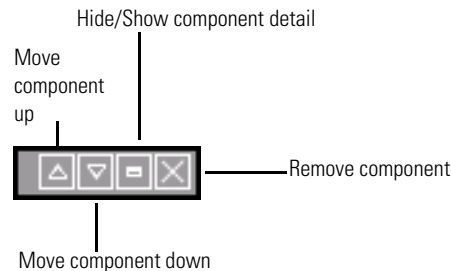
Status Review (10 requests)			
Request Number	Purpose	Approval	Status
REQ001001	asdf	Approved	Submitted
REQ001002	gdf	Approved	Submitted
REQ001003	asdf	Approved	Submitted
REQ001004	asd	Approved	Submitted
REQ001005	asdf	Pending approval	Submitted
REQ001006	asdf	Pending approval	Submitted
REQ001008	asd	Pending approval	Submitted
REQ001009	asdf	Pending approval	Submitted
REQ001014	asdf	Pending approval	Submitted
REQ001020	dfg	Pending approval	Submitted

Changing the layout

The following sections contain procedures for changing the location of the components or removing them from the Peregrine Portal. The procedure you use is determined by the Web browser you are using.

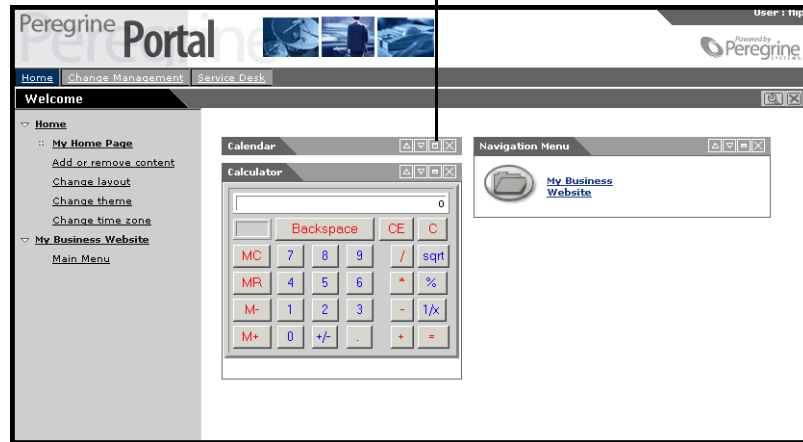
Microsoft Internet Explorer

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.



In the following screen, the Calendar is minimized.

Click the Show/Hide detail button to redisplay hidden components.

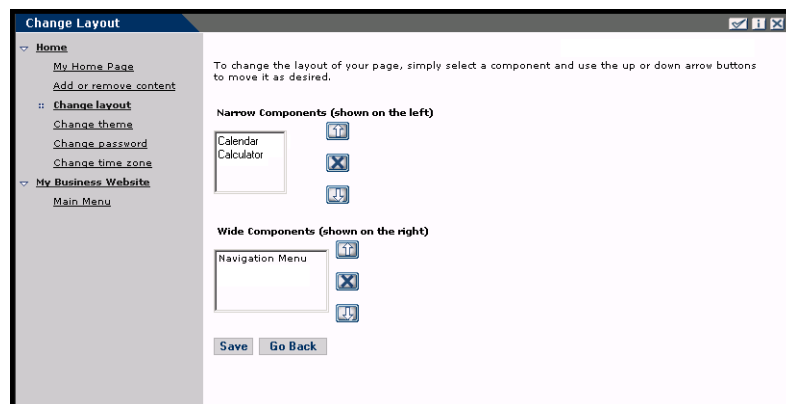


Netscape Navigator

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.

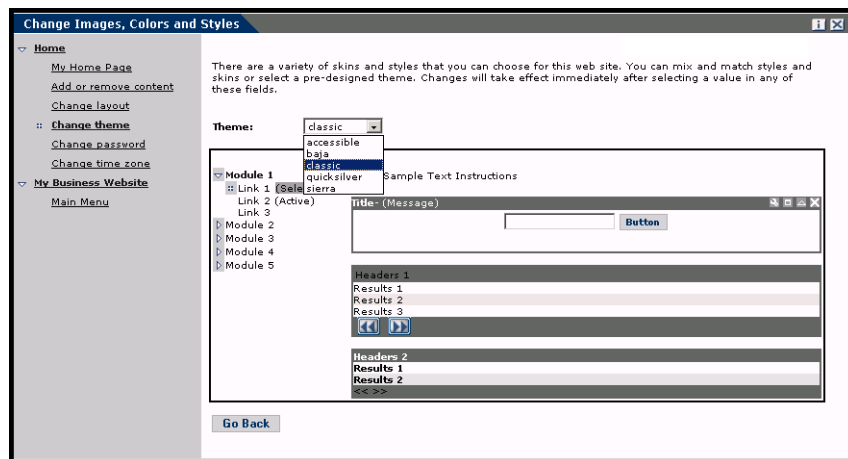
- 2 Select the component you want to modify, and then click the appropriate button to activate the change.
 - Up arrow moves the component up.
 - Down arrow moves the component down.
 - X removes the component from the Peregrine Portal.
- 3 Click Save.

Changing themes

You can choose from a number of themes to change the look of your Web pages. Out of the box, Get-Resources provides five themes you can choose between. If you want to deploy additional themes, refer to *Customizing the Peregrine Portal*.

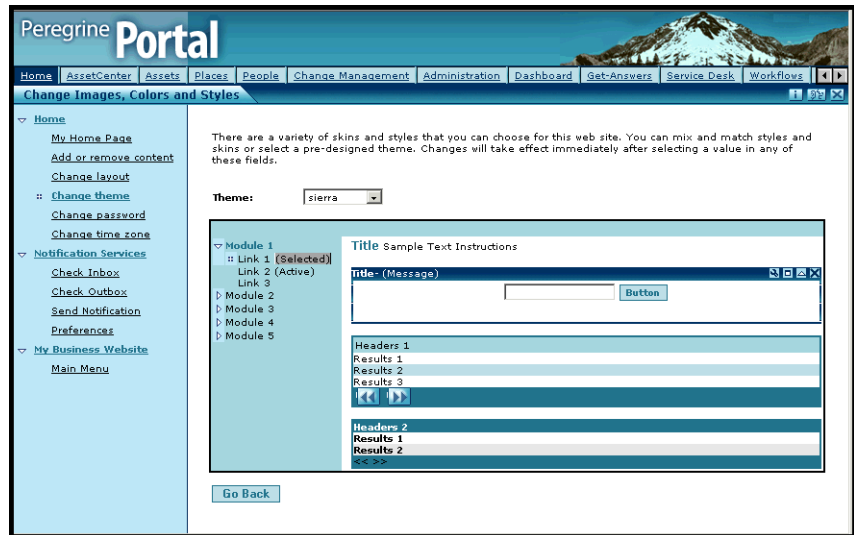
To change the theme:

- 1 From the Activity menu on the Portal Home page, select **Change theme**.
The following page opens.



- 2 Choose 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.



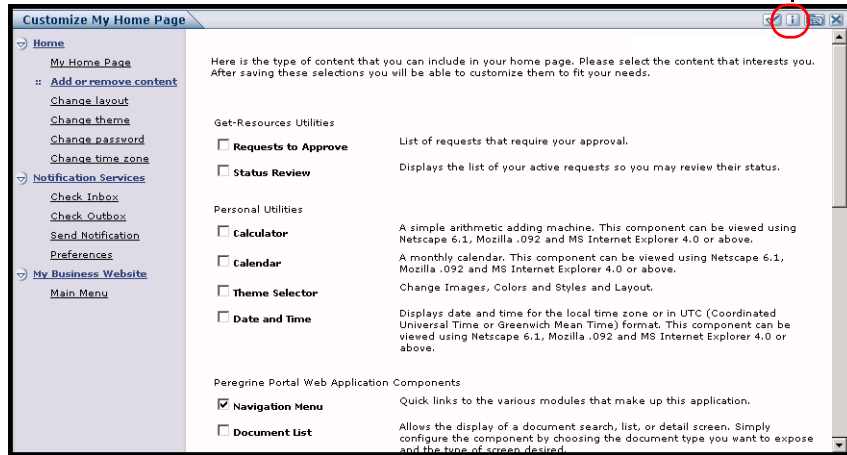
This new configuration remains through subsequent work sessions until changed.

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 Get-Resources 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.



5 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 are displayed, or add, change, or delete records from a back-end database.

This section includes:

- *Personalization overview* on page 66
- *Supporting personalization* on page 74
- *Personalization tasks* on page 77
- *Moving Personalizations from a development to a production environment* on page 82

Personalization overview

Personalization allows end users a means to create and customize searches of Get-Resources 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 Get-Resources interface without the need to rebuild a Peregrine Studio project for every change made. 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. Doc Explorer 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 will appear only in activities where a Personalization form has been defined. The Personalization form determines what options are displayed in the Personalization pop-up window.

When you click on the Personalization icon, a pop-up window opens displaying 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.

Select the fields you want to display when examining a **Ticket** document. Double click on a field in the right column to edit its attributes.

Document Fields	Current Configuration
Available Fields -- Left/Right Split -- -- Top/Bottom Split -- -- Section Title -- Asset Tag Assignment Group Attachments Category Cause Code Company Id Contact Name Description	Current Configuration -- Ticket Details -- Ticket Number Status Category Severity Priority Opened By Open Time -- Top/Bottom Split -- -- Description -- Description

Form Options

Title:

Instructions:

Explorer Options

Create: Go directly to the create screen by default

Skip search: Skip the search page and execute a default query

Single Detail: Go directly to detail page when search finds exactly one item

Summary: Show a summary page for the document

Restrict operations to the following roles:

Document Creation:

Document Deletion:

Document Update:

Administrators determine what available data fields display on each form. The wrench icon only appears in activities where a Personalization form has been defined. 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 subdocument 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.
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 Get-Resources administrator or the form's schema.
Save	Saves and applies your Personalization changes to the current form.





Adding and removing personalizations

You personalize the Get-Resources 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
	<p>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.</p> <p>See the following sections for more information: <i>Configuring fields</i> on page 70; <i>Configuring subdocuments</i> on page 71; and <i>Configuring collections</i> on page 73.</p>
	<p>The Move (vertical arrows) icons move the component either up or down on the page. Moving components is always performed with these arrows.</p>
	
	<p>The Remove (X) icon removes the component from the page. Removing a component does not delete the component; it only does not display it.</p>

Configuring fields



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

[Project.studio.personalize.attribute.start](#) ⓘ
Please change any of the following attributes so that it is presented to meet your exact needs.

cols:	<input type="text"/>
colspan:	<input type="text" value="1"/>
default:	<input type="text"/>
label:	<input type="text" value="Brand"/>
readonly:	<input checked="" type="radio"/> Yes <input type="radio"/> No
required:	<input type="radio"/> Yes <input checked="" type="radio"/> No
rows:	<input type="text" value="1"/>
rowspan:	<input type="text" value="1"/>
size:	<input type="text"/>

See the section *Configuring field attributes* on page 78 for more information about specific tasks. Depending on your personalization rights, you can rename the field label (see page 79), change the field to read-only (see page 79), require users to enter a value (see page 80), and change the size and span of a field (see page 80).

Configuring subdocuments



After you select a subdocument in the Current Configuration list, click the wrench icon to configure the 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 Columns	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
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	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 for displaying the returned value from the lookup activity. If you do not want the lookup icon and link to appear at all, uncheck **Lookup**, clear all checkbox 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 Columns	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	Defines how Peregrine OAA displays results.
Lookup	Users see the Explorer Options section only if they have administrative Personalization rights.
Create	
Popup	
Remove	
Max. Row Count	
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 or ServiceCenter 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 Get-Resources Administration settings page:
 - Portal
 - PortalDB
 - Web Application

Activating Personalization

Personalization is intended to be 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:

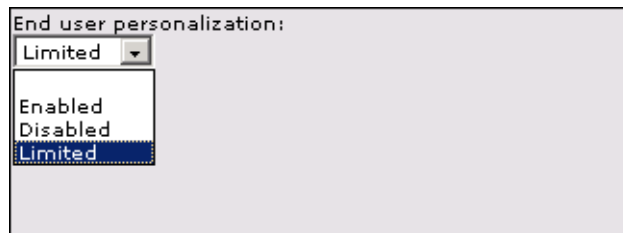
- Grant all users personalization rights by setting the end-user personalization administrative setting.
- Grant individual users personalization rights by adding a capability word to their user profile.

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 Get-Resources 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.



- **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 will be 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 Get-Resources back-end database. The personalization wrench icon will be hidden from the Get-Resources interface, and any end-users who have individual personalization rights will 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 Get-Resources back-end database. The following personalization capabilities words are available:

- `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 Get-Resources 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. 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 Get-Resources back-end database or set global personalization rights.

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. 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 will appear 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 79 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
Colspan	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 Get-Resources 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.
Rowspan	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 existing label of a field, you must have personalization rights.

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**.

Making a schema visible to BVA portal components

The Business View Authoring (BVA) tools – Document List and My Menu – use public schemas to determine what back-end database fields and tables users can see. The Business View Authoring tools can only see the fields and tables that you define in public schemas.

To make a schema visible to portal components:

- 1 Login in to the server where you have installed Get-Resources.
- 2 Open Windows Explorer and navigate to your Get-Resources apps folder. For example:

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

Each module of your Peregrine Studio project has its own folder of schemas.

- 3 Navigate to the folder name matching the module for which you want to enable public schemas. For example:

```
requestincidentmgt
```

- 4 Create a text file called `publicSchemas.xml` in this folder.
- 5 Add the following entries to `publicSchemas.xml`:

```
<schemas>
  <document name="Schema Name" label="Label to appear in BVA"/>
  ...
</schemas>
```

Add one `<document>` element for each schema that you want to make available to the Business View Authoring tools.

For the name attribute, enter the file name of the schema as it is listed in Peregrine Studio.

For the label attribute, enter any text that you want to use to describe the schema. This text will appear as a description in the BVA interfaces.

- 6 Save the text file.
- 7 Repeat step 3 to step 6 for each module that is in your Peregrine Studio project.

Moving Personalizations from a development 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: Only personalizations that were made by users who have `getit.personalization.admin` capability can be exported using the export and import processes.

To export personalization files:

- 1 Log in to the development application server Administration page.
- 2 Click **Administration > Import/Export**.
- 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 Administration page.
- 2 Click **Administration > Import/Export**.
- 3 Change the path and file name to the path and file name of the file you want to import.
- 4 Click **Import**.

6 Document Schema Definitions

CHAPTER

This document describes document schema definitions and explains how they map data between Get-Resources 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* on page 86
- *How to use schemas* on page 88
- *Schema extensions* on page 89
- *Editing the schema extension files* on page 93
- *Schema Subclasses* on page 106
- *Editing the schema subclass files* on page 108
- *Schema Elements and Attributes* on page 118

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 Get-Resources displays and processes.

All schemas consist of two types of definitions:

- **Base definitions**—The schema entries that provide a logical mapping between the XML tags generated in a document query to the Get-Resources 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 118.

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.

```

XML namespace -----<?xml version="1.0"?>
                    <schema>

                    <!--=====
                    Schema extension for logical mappings
                    =====>
Logical mappings always
use name="base" -----<documents name="base">
Document name -----<document name="sample">
determines schema name.   <attribute name="Id" type="number">
This schema is sample.xml <attribute name="contact" type="string" label="Contact" />
                           </document>
                           </documents>
                    </schema>

```

Physical mappings

The file `\schema\extensions\ac\sample.xml` would list the schema extension physical mappings.

```

XML namespace -----<?xml version="1.0"?>
                    <schema>

                    <!--=====
                    Schema extension for physical mappings
                    =====>
Physical mapping lists
adapter name -----<documents name="ac">
                           <document name="sample" table="amRequest">
Physical mapping uses -----<attribute name="Id" field="lReqId" />
same attribute -----<attribute name="contact" field="lEmplDeptId" />
elements -----<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 Get-Resources 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 DocExplorer schemas. 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 90.

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 90.
- Step 2** Locate the schema file on the Get-Resources server. See *Locating the schema on the server* on page 91.
- Step 3** Create the schema extension target folders and copy XML files. See *Creating the schema extension target folders and files* on page 91.
- Step 4** Edit the schema extension files to support the features you want. See *Editing the schema extension files* on page 93.

Identifying the schema to extend

You can identify the schema used by a particular form directly from the Get-Resources 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 Get-Resources 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\resources\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\resources\schema
```

- Schema extension 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\Request.xml.
```

- Schema extension 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\ac\Request.xml.
```

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\resources\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 ac. For example:

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

- 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 93.
- *Hiding an existing field from the Available Fields list* on page 95.
- *Changing the label a field displays in the Available Fields list* on page 96.
- *Changing the list of forms where a field is available or visible* on page 97.
- *Changing the physical mapping of a field* on page 100.
- *Changing the type of form component a field uses* on page 101.
- *Adding subdocuments to the Available Fields list* on page 102.

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:

Add new logical mappings here

```
<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 124 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="ac" version="4">`.
If you cannot find a matching `<documents>` element entry for your adapter, you must create one. See `<documents>` on page 118 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 120 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:

Add new physical mappings here

```
<documents name="ac" 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 124 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 the following four attributes to the `<attribute>` element you want to remove from the DocExplorer Available Fields list:

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

Add search, list, detail,
and create attributes

```
<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.

Update the label attribute

```

<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.

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	<ul style="list-style-type: none"> ■ <code>search=</code> ■ <code>search =true</code> 	<code>search=true</code>	<code>search=false</code>
list	<ul style="list-style-type: none"> ■ <code>list=</code> ■ <code>list=true</code> 	<code>list=true</code>	<code>list=false</code>
detail	<ul style="list-style-type: none"> ■ <code>detail=</code> ■ <code>detail=true</code> 	<code>detail=true</code>	<code>detail=false</code>
create	<ul style="list-style-type: none"> ■ <code>create=</code> ■ <code>create=true</code> 	<code>create=true</code>	<code>create=false</code>

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>

```

Set search, list, detail, and create attributes

- 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="ac" version="4">`.

If you cannot find a matching `<documents>` element entry for your adapter, you must create one. See [<documents>](#) on page 118 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 120 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:

Change physical mappings here

```
<documents name="ac" 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 124 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 124.

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.

```
Update the type attribute——<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 132.

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
<code><document></code>	Use if the subdocument query always returns <i>one and only one</i> result for each requested element in the subdocument. For example, a contact should only have one name.	Required attributes <ul style="list-style-type: none"> ■ name Optional attributes <ul style="list-style-type: none"> ■ docname
<code><collection></code>	Use if the subdocument query can return <i>more than one</i> result for each requested element in the subdocument. For example, a contact can have multiple requests open in his name.	Required attributes <ul style="list-style-type: none"> ■ name Required elements <ul style="list-style-type: none"> ■ <code><document></code>

```

Subdocument with one
result – address —————<documents name="base">
    <document name="schema">
        <attribute name="contact" type="string" label="Contact" />
        ...
        <document name="address" docname="external_schema" />
        ...
Subdocument with
multiple results –
telephone numbers —————<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.

- 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="ac" version="4">`.
If you cannot find a matching `<documents>` element entry for your adapter, you must create one. See [<documents>](#) on page 118 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 fields require, you must edit the attributes. See [<document>](#) on page 120 for more information on the requirements of a `<document>` physical mapping.

- 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 <i>one and only one</i> result for each requested element in the subdocument. For example, a contact should only have one name.	Required attributes <ul style="list-style-type: none"> ■ table ■ field ■ joinfield ■ joinvalue Optional attributes <ul style="list-style-type: none"> ■ docname
<collection>	Use if the subdocument query can return <i>more than one</i> result for each requested element in the subdocument. For example, a contact can have multiple requests open in his name.	Required attributes <ul style="list-style-type: none"> ■ name Required elements <ul style="list-style-type: none"> ■ <document>

```

<documents name="ac" 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>

```

Subdocument maps to external table – table2

Subdocument maps to external table – table3

- 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 106.
- Step 2** Create a `package.xml` file to add your custom files to your Get-Resources installation. See *Creating a package.xml file* on page 107.
- 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 107.
- 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 108.
- Step 5** Create the custom loadscript used by your schema subclass. See *Editing the loadscript files* on page 109.

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 Get-Resources 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 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 Get-Resources 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>
```

Enter an existing
schema name for the
extends attribute

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\schemas\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* on page 109
- *Filtering a list of documents in a field lookup* on page 110
- *Adding data validation for document updates or inserts* on page 112
- *Adding default values to a detail form* on page 114
- *Changing document data when a particular condition is met* on page 116

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(DOEXPLORER_CONTEXT), msg.get(DOEXPLORER_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;
}
```

This line calls the ContactName field of the schema subclass and sets it to the value of the current user name

- 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 Get-Resources 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;
}
```

These lines determine when a lookup field is querying the ticket category

These lines change the default query to add an additional criteria. Category must have a company value of DEFAULT

These lines return the filtered message in a document called fieldlookup

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 Get-Resources 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;
    }
  }

  ...
}
```

These lines read the value of the NewUpdates field and check for the word “password”

These lines displays an error message and return the user to the detail form

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 line obtains the contact record for the user who is currently logged in

This line queries for all the values in the tickets_by_contacts schema subclass

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 a particular condition is met

You can create a loadscript that checks for a particular condition and changes the data in a form before it is submitted to the Get-Resources 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 a particular condition is met:

- 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 line obtains the contact record for the user who is currently logged in

These lines set the Priority field to 1 (using the sPrio variable) when the Department field has the value of Executive

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

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

Optional attributes

- *None*. There are no optional attributes for the logical mapping portion of the schema.

Logical mappings always use name="base" —————

```
<?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

- `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

- `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.

You can add a `<documents>` element for each adapter

```
<?xml version="1.0"?>
<schema>
  ...
  <documents name="acsc" version="34">
    ...
  </documents>
  <documents name="acsc" version="45">
    ...
  </documents>
  ...

```

Each `<documents>` element can describe a different version

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

- **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

- **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 New 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 attribute 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 attribute 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.
- **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 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 function runs when this schema is used in a DocExplorer form. The value of this attribute must be the Peregrine Studio name of the ECMAScript function 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.
- **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.
- **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 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

- **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

- `attachable`. This attribute identifies the ServiceCenter table where references to attachments are located. The value of this attribute must be the SQL name of ServiceCenter table you want to use.

Note: You can only use this attribute when you are using ServiceCenter as your back-end database.

- `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.
- `insert`. This attribute identifies the event name to be sent to ServiceCenter when Get-Services inserts (creates) a new record. The value of this attribute must be the SQL name of the ServiceCenter event.

Note: You can only use this attribute when you are using ServiceCenter as 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.
- **preprocess.** This attribute determines what ECMAScript function 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 function 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.
- **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.
- **update.** This attribute identifies the event name to be sent to ServiceCenter when Get-Services updates an existing record. The value of this attribute must be the SQL name of the ServiceCenter event.

Note: You can only use this attribute when you are using ServiceCenter as your back-end database.

<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

- **name.** This attribute 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 Get-Resources renders the data in the user interface. The value of this attribute must be one of the following strings:
 - **attachment**—This element is a path and file name to an attachment. Get-Resources renders this element as a collection of attachment controls.
 - **boolean**—This element is a true or false string. Get-Resources renders this element as a check box.
 - **date**—This element is a date listing. Get-Resources renders this element as a date edit control that includes a popup calendar.
 - **datetime**—This element is a combined date and time listing. Get-Resources renders this element as a time edit control.
 - **id**—This element is a number that uniquely describes a back-end database record. Get-Resources renders this element as a single-line edit field.
 - **image**—This element is an image. Get-Resources renders this element as an imagefield.
 - **link**—This element is a subdocument described elsewhere in the schema. Get-Resources renders this element a lookup field.
 - **memo**—This element is a text string. Get-Resources renders this element a multi-line edit box.
 - **money**—This element is a currency amount. Get-Resources renders this element a money field that includes a currency selection tool.
 - **number**—This element is an integer. Get-Resources renders this element an editfield with spinner buttons.
 - **preload**—This element is an executable script. Get-Resources runs the script listed in this element.
 - **string**—This element is text. Get-Resources renders this element an editfield.

- `time`—This element is a time listing. Get-Resources renders this element as a time edit control.
- `url`—This element is a Web site address. Get-Resources renders this element as an HREF link icon.

Note: The Archway Document Manager does not validate that the contents of an element matches the type attributed listed for it.

Optional attributes

- `access`. This attribute determines if the element is read-only or editable in DocExplorer forms. 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 make the element editable.
- `ACLcreate`. This attribute determines the default access control list for DocExplorer forms that use this element. The value of this attribute 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 attribute 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 attribute 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 attribute 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.** This attribute determines if the 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 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.
- **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

- **name.** This attribute 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.

Optional attributes

- **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>
```

- **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.

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 requests that a particular user has open. 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

- **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

- **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 attribute 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 attribute 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.
- **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

- **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

- *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:

- **Schema logical definitions.** The schema logical definitions determine what XML elements make up the generated document.
- **The 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.** ECMAScript functions 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 interface.

You can see the raw Get-Resources XML documents by enabling the **Show form information** option from the Administration settings. The form information window displays the following document information:

- **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 service form will invoke a database query through the incident 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 Get-Resources 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 show 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 request, but each request should only have one contact name.

Schema

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

Logical mapping for subdocument – EndUser	—————	<pre> <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> </pre>
Physical mapping for subdocument – EndUser	—————	<pre> ... <document name="EndUser" docname="Employee" table="amEmplDept" field="lUserId" link="lUserId" joinfield="lEmplDeptId" joinvalue="EndUserId"/> ... </document> </documents> </pre>

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.

Elements from schema mapping – Id, Number	—————	<pre> <Request> <Id>32097</Id> <Number>REQ000042</Number> <Purpose>Purpose 1</Purpose> ... <EndUserId>15630</EndUserId> ... </Request> </pre>
Joinvalue – EndUserId	—————	<pre> ... <EndUserId>15630</EndUserId> ... </Request> </pre>

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 requests 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>

```

Logical mapping for subdocuments – RequestLine

No physical mapping for subdocuments – RequestLine. Therefore, physical mapping defaults to that listed in RequestLine schema

```

  <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>

```

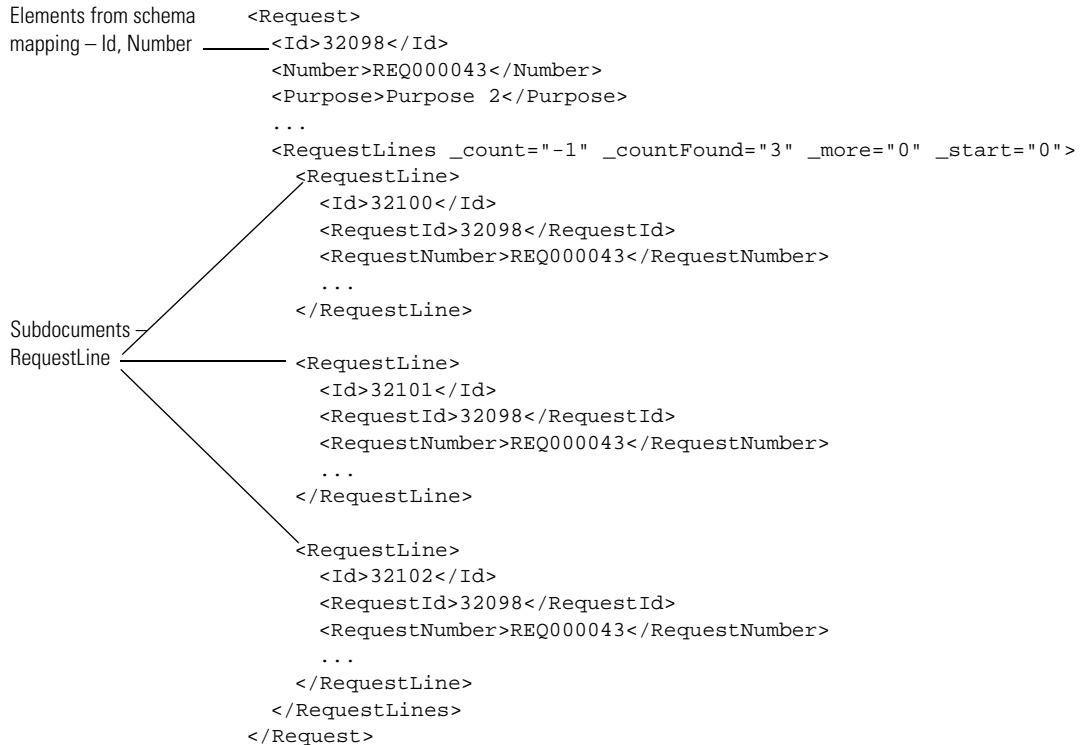
Logical mapping for RequestLine schema

Logical mapping for subdocuments – RequestLine

Physical mapping for subdocuments – RequestLines

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.



7 Modifying the Request Type and Item Type Selection Menus

CHAPTER

This section describes how to customize forms used in Get-Resources 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:

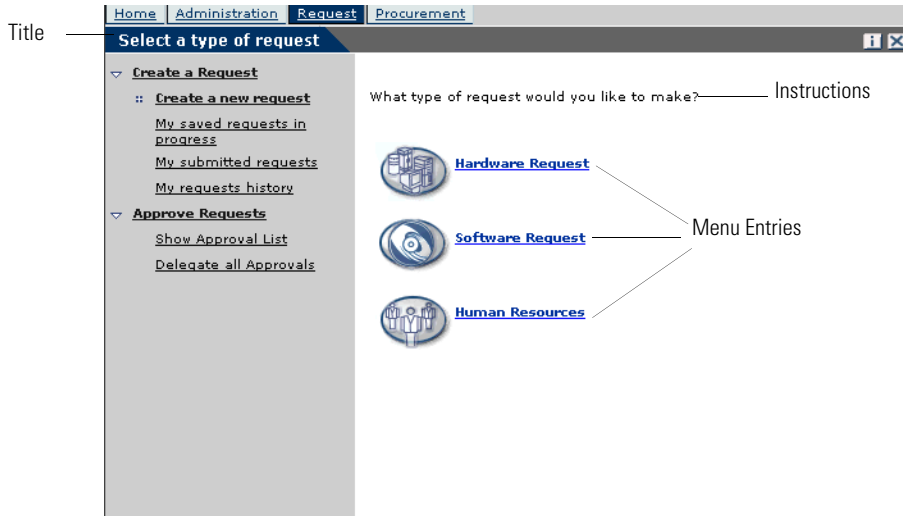
- *Configuring the hierarchical menu component* on page 140
- *Configuring the request type selection menu* on page 147
- *Configuring the item type selection menu* on page 149

Configuring the hierarchical menu component

You can configure the menu forms used in Get-Resources to select the request types and to select the line item types 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.

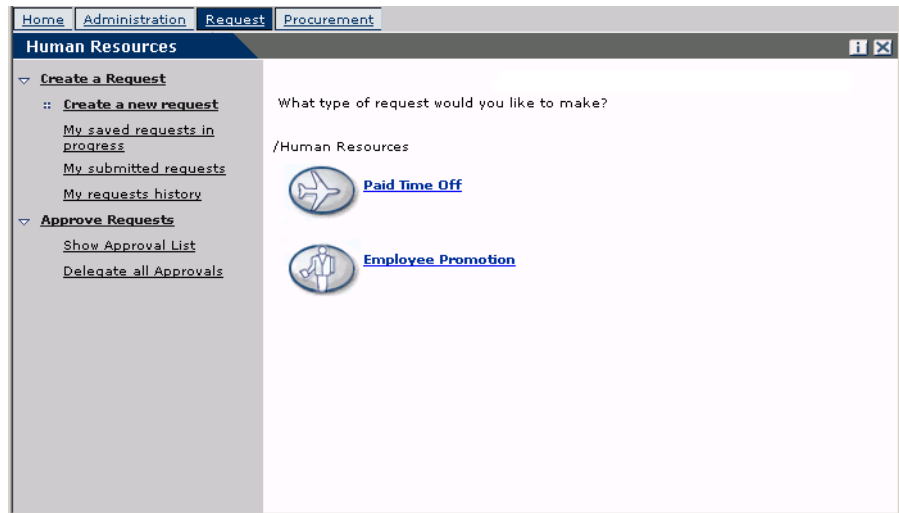
Each menu entry:

- 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 Get-Resources, 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 Get-Resources configuration files are in `WEB-INF\etc\grtrees`.

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.

- **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.
- **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.

Note: 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:

- **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

- **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/aaa_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.
- **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.

- **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.

- **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/oaas_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.
- **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

- **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.
- **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.

Configuring the request type selection menu

Get-Resources ships with default request type menu configuration files that demonstrate some of the options available:

- `WEB-INF\etc\grtrees\acrequestcategory.xml` is a request type selection menu for `AssetCenter`.
- `WEB-INF\etc\grtrees\screquestcategory.xml` is a request type selection menu for `ServiceCenter`.
- `WEB-INF\etc\grtrees\acporequestcategory.xml` is a purchase order type selection menu for `AssetCenter`, when used with the `Procurement` module. The out-of-box configuration is empty.

In the `WEB-INF\etc\grtrees\usersamples` directory, a menu definition file called `acrequestcategory_nocategory.xml`, defines an empty menu. To use this menu definition, copy this file under `WEB-INF\etc\grtrees\user`, and rename it into `acrequestcategory.xml`.

If you want to use your own menu definition for this form, always provide your version and save it as

`WEB-INF\etc\grtrees\user\acrequestcategory.xml` for AssetCenter and `WEB-INF\etc\grtrees\user\screquestcategory.xml` for ServiceCenter.

The files located in the `WEB-INF\etc\grtrees\user` folder are always used instead of the out-of-box files located in `WEB-INF\etc\grtrees`.

Warning: Never modify directly the files that are distributed with the software.

The syntax is an extension of the one described in *Configuring the hierarchical menu component* on page 140. Its syntax is described in the W3Cschema (XML schema): `WEB-INF\etc\grtees\menurequesttype.xsd`.

For this configuration file, the Answers element supports an additional subelement: WizardGRTree.

A new menu entry: the WizardGRTree element

This element has the same subelements as the WizardTarget element, but allows specifying the following subelements that are key to control the behavior of the Get-Resources interface.

- **SubType:** Optional; this element contains an alphanumeric string. If set, the personalization performed on the request summary screen for this request type is saved separately. Any two request types that have the same SubType share the same personalization of the request summary screen.
- **GoToSummary:** Optional; this Boolean element, when set to true, forces Get-Resources to go directly to the request summary screen after the current request type is selected. When absent, this element is considered false, and the next screen is the line item category selection.

- **CanAddMoreItem**: Optional; this Boolean element, when set to false, changes the behavior of the request summary screen so that it is not possible to add more items or go back to the start of the catalog. When absent, this parameter is considered true.
- **CanShowItems**: Optional; this Boolean element, when set to false, hides the list of line items on the request summary page, and hides also the total price.

Configuring the item type selection menu

Get-Resources ships with default item type menu configuration files that demonstrate some of the options available:

- **WEB-INF\etc\grtrees\ac3reqlineitemcategory.xml** is the item type selection menu for AssetCenter 3.x. Note that it imports parts of its definitions from **ac3lineitemcategory.xml**.
- **WEB-INF\etc\grtrees\ac4reqlineitemcategory.xml** is the item type selection menu for AssetCenter 4.x used when the **Procurement Available** option is set to Yes. Note that it imports parts of its definitions from **ac4lineitemcategory.xml**.
- **WEB-INF\etc\grtrees\ac4prreqlineitemcategory.xml** is the item type selection menu for AssetCenter 4.x used when the **Procurement Available** option is set to No.
- **WEB-INF\etc\grtrees\screqlineitemcategory.xml** is the item type selection menu for ServiceCenter. Note that this menu is defined dynamically using Get-Resources specific scripts.
- **WEB-INF\etc\grtrees\acpolineitemcategory.xml** is the item type selection menu for the Get-Resources Procurement module available with AssetCenter. Note that by default it contains only one entry, and this entry is automatically selected. This is why when the user clicks **Create a new PO**, the first screen to appear is directly a list of request lines.

In the `WEB-INF\etc\grtrees\usersamples` directory contains two item type menu configuration files:

- `ac3lineitemcategory_standard.xml` defines the old Get-Resources line item type selection menu based on the Certification field. Note that you are free to use any other field available in the Product schema to determine your different item types.
- `ac4lineitemcategory_standard.xml` defines the line item type selection menu that was available in Get-Resources 4.0.

To use one of these menu definitions, copy the desired file under `WEB-INF\etc\grtrees\user`, and rename it into respectively `ac3lineitemcategory.xml` or `ac4lineitemcategory`.

If you want to use you own menu definition for this form, always provide your version and save it as:

- `WEB-INF\etc\grtrees\user\ac3reqlineitemcategory.xml` for AssetCenter 3.x.
- `WEB-INF\etc\grtrees\user\ac4reqlineitemcategory.xml` for AssetCenter 4.x used when the **Procurement Available** option is set to Yes.
- `WEB-INF\etc\grtrees\user\ac4prreqlineitemcategory.xml` for AssetCenter 4.x used when the **Procurement Available** option is set to No.
- `WEB-INF\etc\grtrees\user\screqlineitemcategory.xml` for Get-Resources for ServiceCenter.
- `WEB-INF\etc\grtrees\user\acpolineitemcategory.xml` for the purchase order line item categories.

Warning: Never modify directly the files that are distributed with the software.

The syntax is an extension of the one described in *Configuring the hierarchical menu component* on page 140. Its syntax is described in the W3C schema (XML schema): `WEB-INF\etc\grtees\menulineitemtype.xsd`.

For this configuration file, the Answers element supports an additional subelement: `WizardGRTree`.

A new menu entry: the WizardGRTree element

This element has the same subelements as the WizardTarget element, but allows specifying the following subelement.

- **DestType:** This optional element can be configured with two values:
 - **list:** Clicking this menu entry presents a list of catalog items available to add to your request. The database schema, and therefore the table used to retrieve the data, depends on the CatalogId.
 - **search:** Clicking this menu entry presents a search page.
- **CatalogId:** This optional value is the name of the catalog script used to retrieve the catalog data from the database. There are three catalogs:
 - **catalogbase:** The default value (used when **CatalogId** is not specified). **WEB-INF/apps/resources/schema/Product.xml** is the database schema used for this catalog. For AssetCenter 3.6, the schema maps to **amProduct** table. For AssetCenter 4.x, the schema maps to **amCatRef** table, and for ServiceCenter, the schema maps to model file.
 - **ac4modelcatalog:** Only available for use with AssetCenter 4.x. **WEB-INF/apps/resources/schema/ac4model.xml** is the database schema used to retrieve the values and it is mapped to the **amModel** table.
 - **ac4bundlecatalog:** Only available for use with AssetCenter 4.x. **WEB-INF/apps/resources/schema/ac4bundle.xml** is the database schema used to retrieve the values and it is mapped to the **amRequest** table.
 - **offcatalog:** Clicking on an icon with this **CatalogId** opens a window from which the user can request an item in free text format.
- **QueryParam:** This optional element represents the search parameters that are used to filter the list of catalog items. The actual search parameters that can be used depend on the database schema that the **CatalogId** uses. This element contains one or more subelements. The name of these subelements are the attribute name (found in the schema), and the value set (used in the query that retrieves the catalog items).

8 Get-Resources Administration

CHAPTER

This chapter includes instructions for administering your Get-Resources system.

Topics in this chapter include:

- *Accessing the Peregrine Portal Admin module* on page 154
- *Using the Control Panel* on page 156
- *Viewing the Deployed Versions* on page 157
- *Using the Settings page* on page 158
- *Logging* on page 160
- *Verifying Script Status* on page 166
- *Displaying Message Queues* on page 166
- *Showing Queue Status* on page 167
- *Importing and exporting personalizations* on page 168
- *Viewing adapter transactions* on page 168
- *Using the IBM Websphere Portal* on page 169
- *Displaying form information* on page 169
- *User self-registration* on page 172
- *Changing passwords* on page 173
- *Logging and monitoring user sessions* on page 173

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 Get-Resources, you must log in as a ServiceCenter or AssetCenter user with `getit.admin` rights to access the Admin module and administer the Get-Resources integration with ServiceCenter or AssetCenter. For a list of access capability words and Adapter configuration instructions, see the section on Get-Resources 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. If you reinstall Get-Resources, make a copy of this file and store it outside your Get-Resources installation. Failure to do this will result in your parameter values being lost during the new installation.

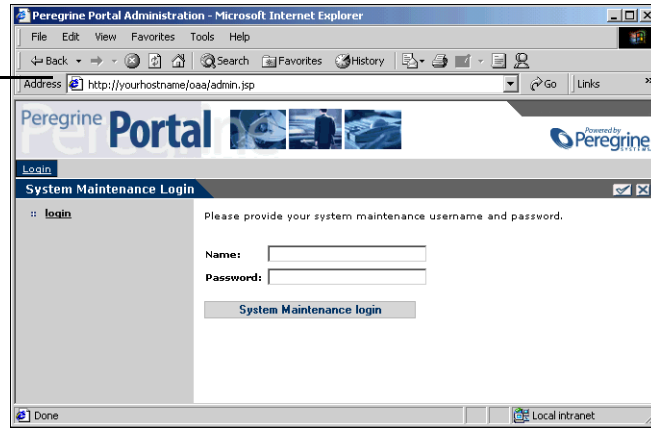
To access the Peregrine Portal administrator login page:

- 1 Verify that your application server (for example, Tomcat) is running.
- 2 In your Web browser Address field, type:
`<hostname>/oaa/admin.jsp`

3 Press Enter to open the Portal administrator login page.

Type your hostname to connect to your local server.

System is the default administrator name.



4 In the Name field, type System.

No password is required on initial login.

5 Click System Maintenance login to open the Control Panel page.

Administrators use the Admin module to define settings to the system.

Target	Adapter	Status
webapplication	com.peregrine.oaa.adapter.scSCAdapter	connected
mail	com.peregrine.oaa.adapter.mail.MailAdapter	connected
portalDB	com.peregrine.oaa.adapter.scSCAdapter	connected
sc	com.peregrine.oaa.adapter.scSCAdapter	connected

Server Name	Last Min.	5 Min. Avg.	20 Min. Avg.	Peak
localhost	1	1	1	2

Server Name	Last Min.	5 Min. Avg.	20 Min. Avg.	Peak
localhost	0	0	0	15

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 Get-Resources server.
Settings	View and change settings for the Peregrine Portal.
Show Script Status	View and verify which 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.
Import / Export	Move Personalizations from a development to a production environment.
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

Using the Control Panel

Use the Control Panel page to check the status of the connections to the databases you are accessing with Get-Resources 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 Server**.

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

Informational, warning, and error messages appear at the top of the page.

The screenshot shows the Control Panel Admin interface. A red circle highlights a message: "The Archway servlet and its Adapter connections have been reset successfully." Below the message is a table titled "Connection Status" showing the status of various adapters.

Target	Adapter	Status
weblocation	com.peregrine.aaa.adapter.sc.SCAAdapter	connected
mail	com.peregrine.aaa.adapter.mail.MailAdapter	connected
portalDE	com.peregrine.aaa.adapter.sc.SCAAdapter	connected
sc	com.peregrine.aaa.adapter.sc.SCAAdapter	connected

Below the connection status table are two summary tables: "Active User Sessions" and "Page Hits per Minute".

Server Name	Last Min.	5 Min. Avg.	20 Min. Avg.	Peak
localhost	1	1	1	2

Server Name	Last Min.	5 Min. Avg.	20 Min. Avg.	Peak
localhost	1	0	0	15

A "Reset Server" button is located at the bottom of the page.

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.

Current applications and their versions are available for viewing with the Deployed Versions option.

The screenshot shows the Deployed Versions screen. A red circle highlights the "Deployed Versions" option in the left-hand navigation menu. The main content area displays a list of applications and their versions.

Applications	Versions
Peregrine Enterprise Portal Accessibility Theme	accessibletheme.4.1.0.5
OAA Archway Servlet	archway.4.1.0.27
Peregrine Enterprise Portal Baja Theme	bajatheme.4.1.0.4
Peregrine Enterprise Portal Classic Theme	classitheme.4.1.0.4
OAA Core Application	core.4.1.0.23
Get-Services Change Request	get-services-change-request.4.1.0.16
Get-Services Change	get-services-change.4.1.0.15
Get-Services	get-services.4.1.0.10
Get-IT Common Utilities	getitcommon.4.1.0.10
Mail Adapter	mailadapter.4.1.0.4
Peregrine Enterprise Portal	portal.4.1.0.28
Peregrine Enterprise Portal Quicksilver Theme	quicksilvertheme.4.1.0.4
ServiceCenter Adapter	scadapter.4.1.0.7
Peregrine Enterprise Portal Sierra Theme	sierratheme.4.1.0.5

A "Print" button is located at the bottom of the list.

- 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.

Settings for the Portal, PortalDB, Web Application, AssetCenter (ACadapter) GRRequestDB, and Service Center (SCadapter) tabs are set during the installation (refer to the *Get-Resources Installation Guide*). You can access the Settings page at any time to change the installation settings. Set the E-mail tab only when users have access to self-registration (see *User self-registration* on page 172).

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.

The screenshot shows the 'Admin Settings' interface. On the left, a navigation menu lists 'Admin', 'Control Panel', 'Deployed Versions', 'Server Log', 'Settings' (highlighted with a red circle), 'Show Script Status', 'Show Message Queues', 'Show Queue Status', 'Import / Export', 'Adapter', 'Transactions/Minute', and 'IBM WebSphere Portal Integration'. The main content area has tabs for 'Change Management', 'Common', 'E-mail', 'Logging', 'Portal', 'Portal_DB', 'ServiceCenter', 'Service_Desk', and 'Themes'. The 'Common' tab is active, displaying several configuration fields with descriptions:

- Maximum attached file size (in KB):** 0. Description: The size limit, in KB, of files that may be submitted as attachments. A value of 0 indicates that no limit is set. This setting is a default that can be overridden by individual attachment fields.
- Common Backend:** portalDB. Description: Adapter target name used to support common user operations.
- List of target aliases:** |weblication;mail. Description: Specifies a list of semicolon delimited target aliases used by web applications in this package.
- System Maintenance username:** System. Description: The system maintenance username. This login provides access to administrative functionality. The system maintenance user is independent of any deployed adapter(s). Use this login to configure a newly installed system or to troubleshoot an existing install.
- System Maintenance password:** [empty field]. Description: The system maintenance password.
- Application path:** WEB-INF/apps/. Description: Directory location of the Peregrine Portal Web Applications.
- Event queue:** portalDB. Description: Enter the name of the adapter that should be used by the Peregrine Portal event queue engine. For example:
 - To use ServiceCenter's repository, enter "sc"
 - To use AssetCenter's repository, enter "ac"

Below these settings is a section for 'Language Translation' with the following field:

- Translation Server Factory Class:** com.peregrine.util.WTSLanguageTranslatorFactory. Description: The Java factory class which generates the proper class associated with the Translation Server.

Setting parameters using the Admin module

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

Important: If you change parameters on the Admin Settings page and then need to reinstall Get-Resources, it is important that you copy the `local.xml` file to a location other than your Get-Resources installation, or all of your settings will be lost when you redeploy Get-Resources. After the installation, move the copy back to the `WEB-INF` directory.

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.

The Control Panel opens.

- 3 Click **Reset Server**.

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.

To enable additional login languages:

- 1 Click **Settings** in the Control Panel.
- 2 Scroll down to the **Encoding, Locales, and Sessions** section.

- 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.

- 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`.

Common	Dashboard	DashboardDB	E-mail	Logging	Portal	Portal DB	ServiceCenter	Themes	Web Application	XSL
Logging										
Log domains:					Enter a semicolon-separated list of execution log traces you want to enable. Choices include:					
<input type="text"/>					<ul style="list-style-type: none"> • dll - Adapter DLL loading and unloading • weblication - Web Application and personalization rendering • jvm - Java run-time environment management and status • locks - Script synchronization locks • security - Archway security trace • statistics - administration statistics 					
Debug script:					When enabled, information regarding ECMA Script execution is written to the log. Be sure to turn this off in a production system.					
<input type="radio"/> Yes <input checked="" type="radio"/> No										
Show form info:					When selected, form information is displayed in each screen to aid during Web Application development and customization.					
<input type="radio"/> Yes <input checked="" type="radio"/> No										
Log file:					Enter a full directory path to the file used for logging.					
<input type="text" value="archway.log"/>										
Logging Format:					The logging format controls the printing pattern in a log file. The format is composed of literal text and conversion specifiers. The details of the specifiers can be found in the Apache Log4j documentation.					
<input type="text" value="%d %p [%t] %x - %m%n"/>										
Log Level:					Controls the level of detail in the log file. Possible values are: all, debug, info, warn, error, fatal and off.					
Information										
Log File Rollover Frequency Pattern:					This setting controls the frequency at which the log file is rolled over. The pattern is also used as an extension to name non-active files. By default the log will roll over at midnight on the first day of each week. More information can be found in the Apache Log4j documentation.					
<input type="text" value="! ,yyyy-ww"/>										
Log Viewer Maximum Size:					This sets the maximum number of lines that the Administration log viewer will display.					
<input type="text" value="70"/>										

You can specify the following log traces in the Log domains field:

- dll - Adapter dll loading and unloading
- weblication - Web application and personalization rendering

- jvm - Java run-time environment management and status
- locks - Script synchronization locks
- security - Archway security trace
- statistics - Administration statistics

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	<p>Used to output the category of the logging event. The category conversion specifier can be optionally followed by <i>precision specifier</i>, which is a decimal constant in brackets.</p> <p>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.</p> <p>For example, for the category name "a.b.c" the pattern %c{2} is output as "b.c".</p>
C	<p>Used to output the fully qualified class name of the caller issuing the logging request. This conversion specifier can be optionally followed by <i>precision specifier</i>, which is a decimal constant in brackets.</p> <p>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.</p> <p>For example, for the class name "org.apache.xyz.SomeClass", the pattern %C{1} is output as "SomeClass".</p> <p>Note: Generating the caller class information is slow. Avoid it unless execution speed is not an issue.</p>
d	<p>Used to output the date of the logging event. The date conversion specifier may be followed by a <i>date format specifier</i>, which is enclosed in braces, such as</p> <p>%d{HH:mm:ss,SSS}</p> <p>or</p> <p>%d{dd MMM yyyy HH:mm:ss,SSS}</p> <p>If no date format specifier is given then ISO8601 format is assumed.</p>
F	<p>Used to output the file name where the logging request was issued.</p> <p>Note: Generating caller location information is extremely slow. Avoid it unless execution speed is not an issue</p>

Conversion character	Effect
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 <i>extremely</i> slow. Avoid it unless execution speed is not an issue.
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.

Conversion character	Effect
X	Used to output the MDC (mapped diagnostic context) associated with the thread that generated the logging event. The X conversion character <i>must</i> be followed by the key for the map placed between braces, as in: <code>%X{clientNumber}</code> where <i>clientNumber</i> 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

Log file rollover frequency is determined by the Apache DailyRollingFileAppender class.

Viewing the Server Log

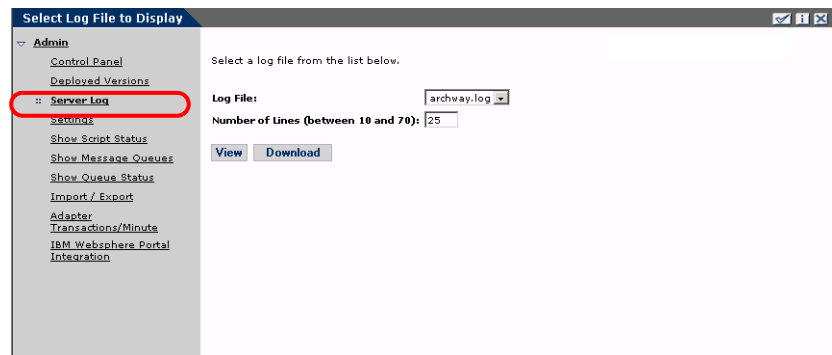
The Server Log provides a history of server events. The default file name is `archway.log`.

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.

You can view the log file from your Web browser or download it to your preferred location.



- 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.



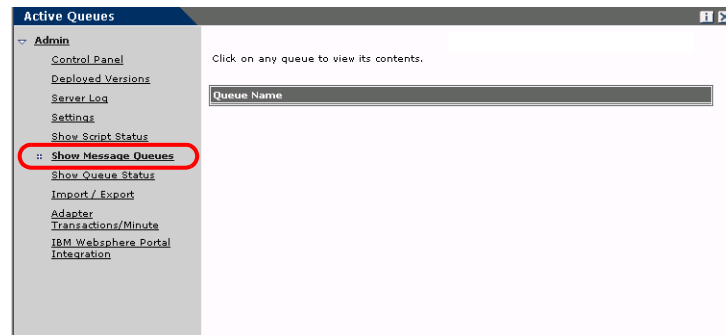
- 2 Click on the script to suspend it.

Displaying Message Queues

The Message Queues are displayed 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.



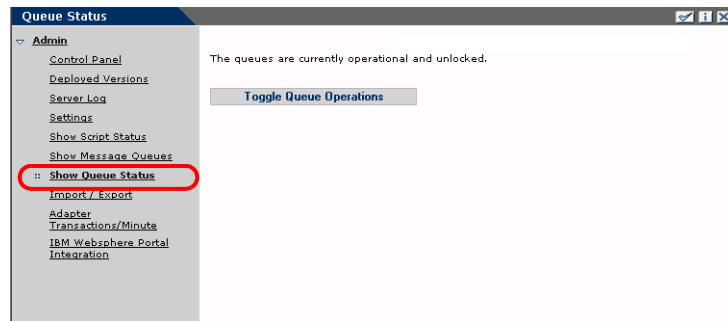
- 2 Click the queue name in the list to view the contents of a queue.

Showing Queue Status

Use the Show Queue Status option to verify or change the status of the message queues.

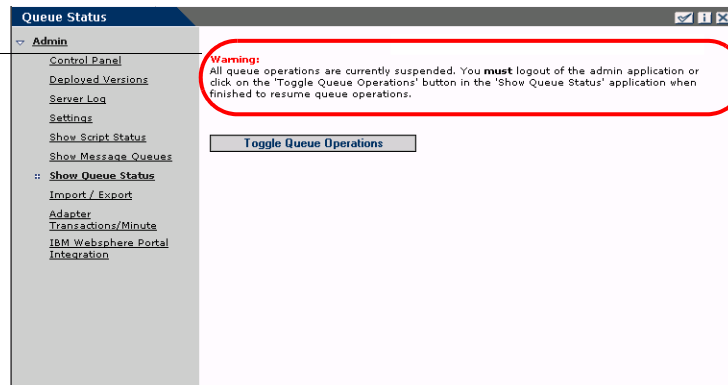
To show queue status:

- 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.

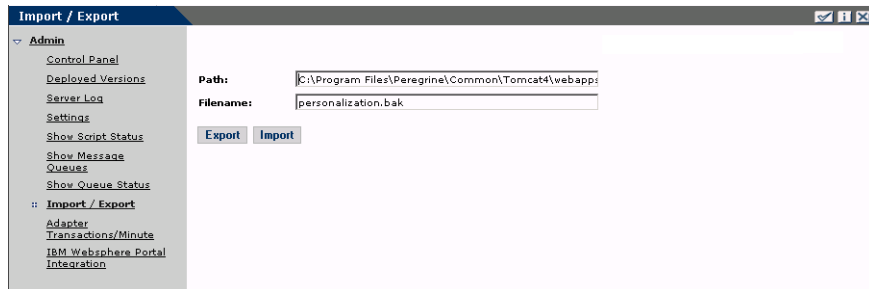
A warning message indicates that the Queue Status is suspended.



- 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.

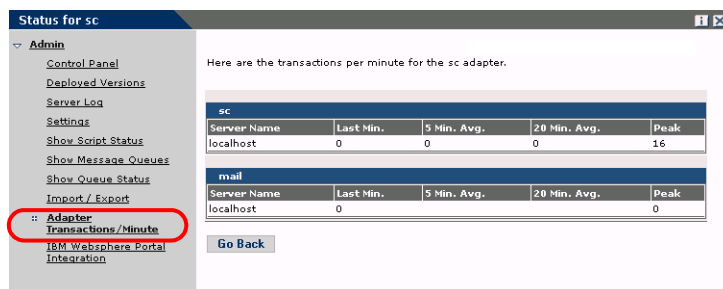


Viewing adapter transactions

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.

IBM Websphere Portal Integration

An IBM Websphere Portal Server web archive configured with references to installed OAA portal components can be generated from this page. The websphere.war file found in the installed packages directory is copied and the portlet.xml file within is replaced. Make sure the base URL is the correct URL for accessing pages on this server. Take the generated file and install it using the IBM WPS Portal Administration utility. Anytime new OAA applications are installed, this process should be repeated to expose any new portal components in the IBM WPS environment.

Source Path: Enter the complete source path on the server where the installed websphere.war file can be located.

Destination Path: Enter the destination path on the server where the generated websphere.war file will be created.

Base URL: Enter the base URL of this server.

Generate WAR File

- 2 Enter the following information:
 - source path
 - destination path
 - base URL
- 3 Click **Generate WAR File**.

Displaying form information

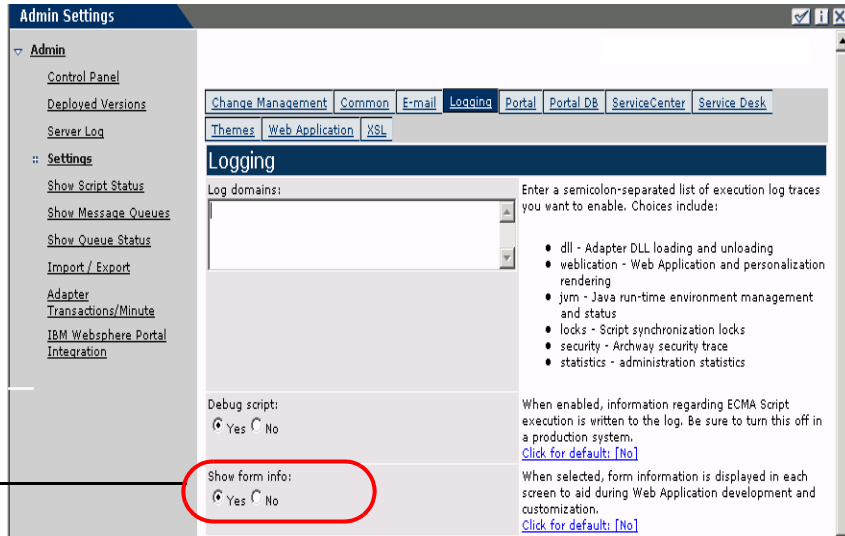
You can use the Admin module to configure Web application forms to display the location and file name of the current form.

To display form information:

- 1 From the Admin module, click **Settings**, then **Logging**.

2 Scroll to the Show form info field, and click Yes if necessary.

Set Show Form Info to Yes.



3 Click Save.

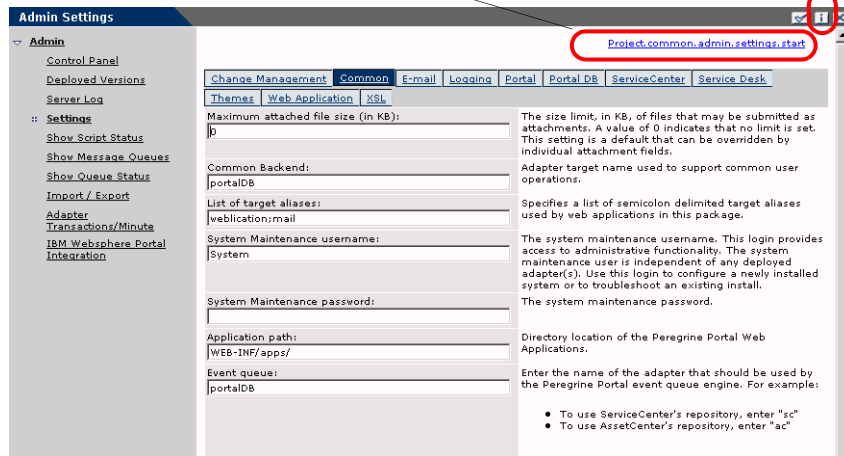
The Control Panel opens.

4 Click Reset Server.

The name of the form is at the top of each form.

The form name is at the top of the page.

Click the Display Form Info button to view the form composition.



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.

View the contents in each tab for more information about the form.

```
<?xml version="1.0" encoding="UTF-8"?>
<_doc>
  <_docExplorerView/>
  <_docExplorerModels>
    <target>portalDB</target>
    <_docExplorerContext>AdminSettings</_docExplorerContext>
    <_docExplorerInstance/>
    <_DocExplorerBackend>webication</_DocExplorerBackend>
    <_docExplorerSubType/>
    <_docExplorerAction>detail</_docExplorerAction>
    <_form>e_admin_settings_start.do</_form>
    <_module>common</_module>
    <_module>admin</_module>
    <_activity>settings</_activity>
    <_formname>start</_formname>
    <_return/>
    <_count>20</_count>
    <_tabs>
      <tab balloon="$$$IDS(common.configPortalLabel)" caption="$$$IDS(common.configPortalLabel)" url="e_admin_settings_start.do?target=portal"/>
      <tab balloon="$$$IDS(common.configCommonLabel)" caption="$$$IDS(common.configCommonLabel)" url="e_admin_settings_start.do?target=common"/>
      <tab balloon="$$$IDS(incidentmgt.configProblemTabLabel)" caption="$$$IDS(incidentmgt.configProblemTabLabel)" url="e_admin_settings_start.do?target=incidentmgt"/>
      <tab balloon="$$$IDS(common.configPortalDBLabel)" caption="$$$IDS(common.configPortalDBLabel)" selected="true" url="e_admin_settings_start.do?target=portalDB"/>
      <tab balloon="$$$IDS(common.configThemesLabel)" caption="$$$IDS(common.configThemesLabel)" url="e_admin_settings_start.do?target=themes"/>
      <tab balloon="$$$IDS(common.configWebicationLabel)" caption="$$$IDS(common.configWebicationLabel)" url="e_admin_settings_start.do?target=webication"/>
      <tab balloon="$$$IDS(common.configLoggingLabel)" caption="$$$IDS(common.configLoggingLabel)" url="e_admin_settings_start.do?target=logging"/>
      <tab balloon="$$$IDS(common.configSLabel)" caption="$$$IDS(common.configSLabel)" url="e_admin_settings_start.do?target=sc"/>
      <tab balloon="XSL" caption="XSL" url="e_admin_settings_start.do?target=xsl"/>
      <tab balloon="$$$IDS(mailadapter.Label)" caption="$$$IDS(mailadapter.Label)" url="e_admin_settings_start.do?target=mail"/>
      <tab balloon="$$$IDS(changerequestmgt.configChangeTabLabel)" caption="$$$IDS(changerequestmgt.configChangeTabLabel)" url="e_admin_settings_start.do?target=changemgt"/>
    </_tabs>
    <_locale>en</_locale>
  </_docExplorerModel>
</_doc>
```

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.
Application Channel Source	HTML source code for the shared applications.

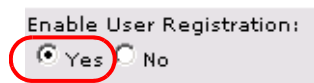
This tab	Contains
Tab Source	HTML source code for tabs.
Menu Source	HTML source code for menus.
Sync/Update Window	HTML source code to synchronize with the page and reload.
Help	Help for debugging the window.

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 ServiceCenter database. When the user registers, ServiceCenter 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 5.0 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.

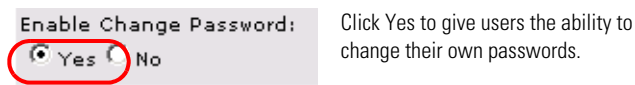
ServiceCenter 5.0 stores names in the format last name/first name. The OAA Platform stores names in the format first name/last name. As a temporary solution, you can change the way operator names are handled in ServiceCenter using the **Use Operator Full Name?** option in the Environment records for Incident and Service Managements. Refer to the *ServiceCenter 5.x Application Administration Guide* 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**.



- 3 Click **Yes**.

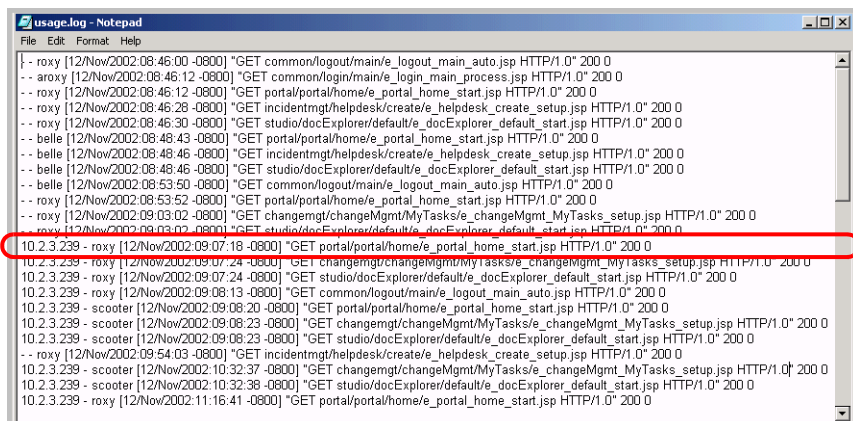
Logging and monitoring user sessions

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:

```
10.2.3.239 - roxy [12/Nov/2002:09:07:18 -0800] "GET
portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
```



```
usage.log - Notepad
File Edit Format Help
{ - roxy [12/Nov/2002:08:46:00 -0800] "GET common/logout/main/e_logout_main_auto.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:08:46:12 -0800] "GET common/login/main/e_login_main_process.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:08:46:12 -0800] "GET portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:08:46:28 -0800] "GET incidentmgmt/helpdesk/create/e_helpdesk_create_setup.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:08:46:30 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
-- belle [12/Nov/2002:08:46:43 -0800] "GET portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
-- belle [12/Nov/2002:08:48:46 -0800] "GET incidentmgmt/helpdesk/create/e_helpdesk_create_setup.jsp HTTP/1.0" 200 0
-- belle [12/Nov/2002:08:48:46 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
-- belle [12/Nov/2002:08:53:50 -0800] "GET common/logout/main/e_logout_main_auto.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:08:53:52 -0800] "GET portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:09:03:02 -0800] "GET changemgt/changeMgmt/MyTasks/e_changeMgmt_MyTasks_setup.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:09:03:02 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
10.2.3.239 - roxy [12/Nov/2002:09:07:18 -0800] "GET portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
10.2.3.239 - roxy [12/Nov/2002:09:07:24 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
10.2.3.239 - roxy [12/Nov/2002:09:08:13 -0800] "GET common/logout/main/e_logout_main_auto.jsp HTTP/1.0" 200 0
10.2.3.239 - scooter [12/Nov/2002:09:08:20 -0800] "GET portal/portal/home/e_portal_home_start.jsp HTTP/1.0" 200 0
10.2.3.239 - scooter [12/Nov/2002:09:08:23 -0800] "GET changemgt/changeMgmt/MyTasks/e_changeMgmt_MyTasks_setup.jsp HTTP/1.0" 200 0
10.2.3.239 - scooter [12/Nov/2002:09:08:23 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
-- roxy [12/Nov/2002:09:54:03 -0800] "GET incidentmgmt/helpdesk/create/e_helpdesk_create_setup.jsp HTTP/1.0" 200 0
10.2.3.239 - scooter [12/Nov/2002:10:32:37 -0800] "GET changemgt/changeMgmt/MyTasks/e_changeMgmt_MyTasks_setup.jsp HTTP/1.0" 200 0
10.2.3.239 - scooter [12/Nov/2002:10:32:38 -0800] "GET studio/docExplorer/default/e_docExplorer_default_start.jsp HTTP/1.0" 200 0
10.2.3.239 - roxy [12/Nov/2002:11:16:41 -0800] "GET portal/portal/home/e_portal_home_start.jsp 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
10.2.3.239	-	roxy	[12/Nov/2002:09:07:18 -0800]	"GET portal/portal/home/e_portal_home_start.jsp 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. In the previous above, the module accessed is <i>notificationsservices</i> , the location of the login script.
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

CHAPTER

The following sections describe administrative functions that must be performed outside of Get-Resources to support certain features available with ServiceCenter or AssetCenter:

- *Configuring the Purchase Order Generation workflow* on page 175
- *Configuring the Product Catalog* on page 177
- *Installing and Configuring the ACAdapter on UNIX* on page 178

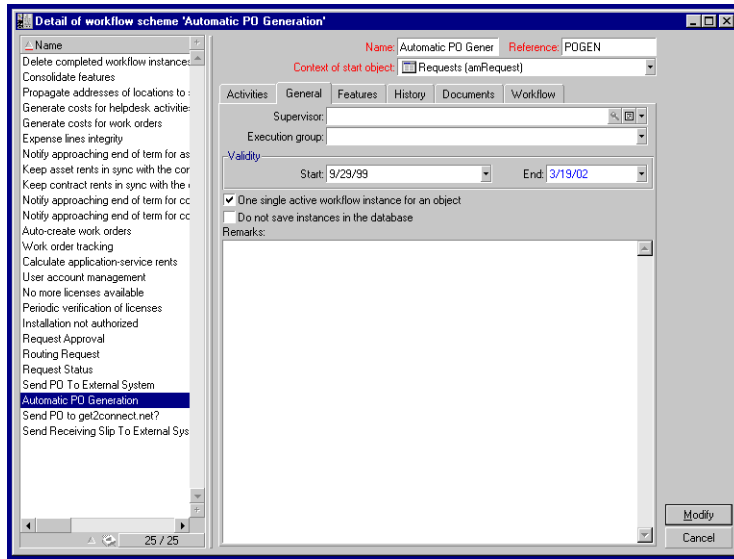
Configuring the Purchase Order Generation workflow

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

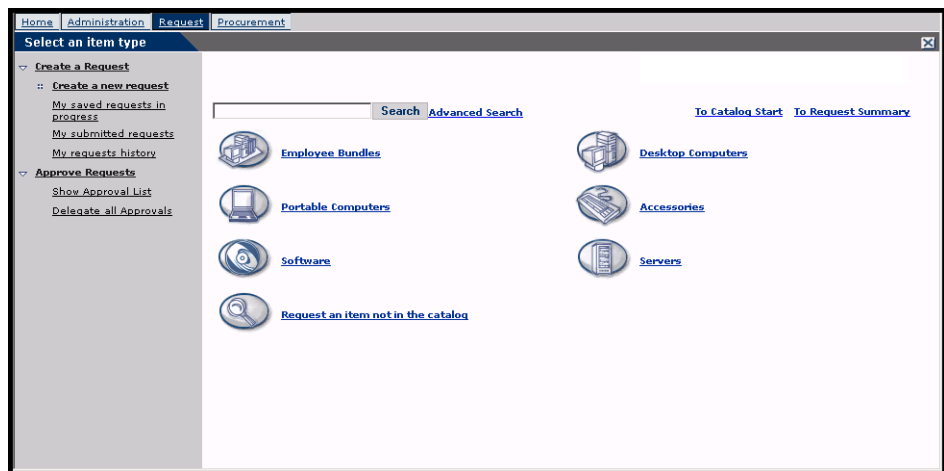
This information applies only to AssetCenter 3.6.

Get-Resources 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

Get-Resources uses the Certification field to determine the availability of items in the AssetCenter catalog.



These buttons from the Get-Resources menu each drive a database call against the amProduct table. The queries executed are similar to the following for Desktop Computers:

```
SELECT IProdId,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 weblocation. 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 Get-Resources 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 Get-Resources 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 If necessary, follow the AssetCenter documentation to install AssetCenter on the AIX, Linux, or Solaris system where OAA will be running.

Note: AssetCenter 4.2 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.

- 8 Optionally, set the AssetCenter library path to your **AssetCenter/bin** directory, including a trailing slash.
- 9 Click **Save**.

10 Security

CHAPTER

This chapter describes the different security configuration options available in Get-Resources. Topics in this chapter include:

- *Password encoding methods* on page 182
- *Back-end system security* on page 183
- *Get-Resources global access rights* on page 187
- *User registration* on page 188
- *Authenticating users* on page 190
- *Default security configuration* on page 191
- *Custom JAAS configuration* on page 192
- *Standard Sun Microsystems JAAS configuration* on page 200
- *Integrated Windows Authentication* on page 201
- *Integrating with single sign-on tools* on page 210
- *Contact-based authentication* on page 215
- *AssetCenter authentication* on page 222
- *Creating an alternate login page* on page 223

Password encoding methods

By default, Get-Resources does not encode passwords sent over the network. Get-Resources 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 Get-Resources passwords, you have three options:

- Enable Secure Sockets Layer (SSL) on your Web server
- Configure Get-Resources to use a directory service such as LDAP
- Enable your Web server to use Windows NT Challenge/Response

In order to use SSL, you need to acquire a digital certificate. If your Web server has a certificate, then your Get-Resources 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.

Get-Resources also supports authentication via a directory service such as LDAP. When you authenticate to a directory service, Get-Resources passes SHA hash encoding passwords to the service. For instructions configuring a directory service see *Custom JAAS configuration* on page 192.

Get-Resources 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 *Integrated Windows Authentication* on page 201.

Back-end system security

This section includes information about how Get-Resources authenticates users and stores personalization changes in the ServiceCenter or AssetCenter back-end system.

Authentication with ServiceCenter or AssetCenter

When a user logs in to Get-Resources, the user name and password are validated against a corresponding **operator record** in ServiceCenter or the Employee table in AssetCenter. Each operator record or employee profile must contain appropriate **capability words** or **user right keywords** in order to gain access to various features within Get-Resources.

Refer to the *ServiceCenter Administration Guide* for more information on operator records, or the *AssetCenter Administration Guide* for more information on profiles.

ServiceCenter capability words and AssetCenter user right keywords

Following is a list of available capability words and user rights keywords for Get-Resources functionality that can be assigned to an operator record in ServiceCenter or a profile in AssetCenter:

ServiceCenter capability word or AssetCenter user right keyword	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.

ServiceCenter capability word or AssetCenter user right keyword	Description
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 Get-Resources interface.
getit.personalization.limited	Can only personalize features that have been exposed by a user with greater personalization rights.
getit.portal	Can view the OAA home page and portal components.
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.
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:

getit.requester
getit.advancedrequester
getit.approver

Refer to the *ServiceCenter Administration Guide* 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

Get-Resources 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.

Get-Resources includes the following sample profiles:

Profile	Privileges
getit.admin	Requesting Approving Purchasing Receiving System Administration PCard Administration
getit.default	Requesting
getit.full	Requesting Approving Purchasing Receiving PCard Administration
Note: AssetCenter Procurement only	
getit.buyer	Requesting Purchasing
Note: AssetCenter Procurement only	
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 Get-Resources:

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.

ServiceCenter password security

You can set the ServiceCenter parameter `securepassword` in the ServiceCenter `sc.ini` file to prevent advanced users from submitting a Get-Resources query that returns a list of user passwords.

To set the password security parameter in ServiceCenter:

- 1 Open the `sc.ini` file in a text editor.
- 2 Add the parameter `securepassword` to the file, and save the file.

A request for a list of passwords in Get-Resources returns a list with the passwords masked.

Get-Resources global access rights

Although initial login access for Get-Resources is validated against the user's corresponding operator record in ServiceCenter or 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` as a global access right gives all users the ability to create requests inaccess Get-Resources even if it were not initially assigned to each user's operator record in ServiceCenter or profile in AssetCenter.

Global access rights are defined on the ServiceCenter or AssetCenter settings page of the Peregrine Portal Administration module.

To define global access rights within Get-Resources:

- 1 Open the Peregrine Portal Administration module in Get-Resources.
- 2 In the left menu pane, click **Settings**.
- 3 On the **Settings** page:
 - Click the **ServiceCenter** tab if ServiceCenter is your back-end system.
 - Click the **AssetCenter** tab if AssetCenter is your back-end system.
- 4 In the ServiceCenter or 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 or `sc` for ServiceCenter as a back-end database.

Multiple default access rights may be granted by separating the capability parameter values with a semicolon (;). For example:

`sc(getit;getit)`

Following is an example of how you update the appropriate settings page field for ServiceCenter or AssetCenter to grant all users the default right to create requests in Get-Resources:

Settings page	Field name	Sample field value
AssetCenter	Default Capability Words:	<code>ac(getit.requester)</code>
ServiceCenter	Default capabilities:	<code>sc(getit.requester)</code>

- 5 Scroll to the bottom of the form and click **Save**.
- 6 When the Control Panel page is displayed, click **Reset Server** to apply your configuration changes.

User registration

All Get-Resources users need a login account in the back-end database providing authentication. For example, if you are using ServiceCenter as your back-end database, then the appropriate capability words must be defined in the user's Operator record. In AssetCenter, the appropriate user rights are defined in the user's Profile. Similar access rights can be defined in any back-end system that you are using. The user login is automatically authenticated in the back-end system.

If a user is attempting to log in for the first time without back-end authentication, the user is prompted for certain default information as shown in the following page. The first four fields are required, as indicated by the arrows to the right of each field.

The screenshot shows the 'Peregrine Portal' user registration interface. At the top, there is a navigation bar with 'Login' and 'Register' links. The main content area is titled 'User Information' and contains a registration form. The form includes the following fields: 'First Name', 'Last Name', 'Login Name', 'Email Address', and 'Phone Number'. Each of the first four fields has a small blue arrow icon to its right, indicating that these fields are required. Below the form is a 'Register' button. A note above the form states: 'You may register online for a new user account. Please provide the requested information. After the account is created, your password will be sent to you via email. Please note that an account can only be created when you provide a valid and authorized company email address.' The browser's address bar at the bottom shows 'Local intranet'.

When the user clicks **Register**, the information is stored in the appropriate database. In AssetCenter, Get-Resources 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. In ServiceCenter, Get-Resources creates an operator and contact record for the new user.

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.

Enabling the E-mail adapter

If users have the ability to self-register, you must make sure that the E-mail tab from the Get-Resources 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.

Portal	Common	Portal DB	Web Application	Logging	XSL	E-mail
Inbound mail host:		The full name or IP address of the machine hosting the inbound mail server. If this field is empty, then the status of the mail adapter will indicate the status of the outbound mail server connection				
<input type="text" value="mailhost"/>						
Inbound mail protocol:		The protocol used by the inbound mail server, which is either imap or pop3.				
<input type="text" value="imap"/>						
Inbound mail user ID:		The user ID used to access the inbound mail server.				
<input type="text"/>						
Inbound mail password:		The user password used to access the inbound mail server.				
<input type="text"/>						
Mail sender address:		This address is used as the default sender address in outbound email messages.				
<input type="text"/>						
Legal domains:		Enter a semicolon-separated list of mail domains that the Peregrine Portal may correspond with. Only users with an email address in these domains are allowed to complete online self-registration.				
<input type="text" value="peregrine.com;apsydev.com;getmark.etaccess.com"/>						
Anonymous user:		Anonymous user name used when an unknown user attempts to communicate with the mail adapter				
<input type="text" value="falcon"/>						
Anonymous password:		Anonymous user password for the mail adapter				
<input type="text"/>						
Outbound mail host:		The full name or IP address of the machine hosting the outbound mail server. Click for default: [mailhost]				
<input type="text" value="condor.peregrine.com"/>						
Outbound mail user ID:		The user ID used to access the outbound mail server.				
<input type="text"/>						
Outbound mail password:		The user password used to access the outbound mail server.				
<input type="text"/>						
Adapter:		Full class path for adapter associated with this target.				
<input type="text" value="com.peregrine.oaa.adapter.mail.MailAdapter"/>						
<input type="button" value="Save"/>						

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 191.
- Use a custom configuration to authenticate users against user-defined adapters such as LDAP or JDBC compliant databases. See *Custom JAAS configuration* on page 192.
- 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 200.

- Use Integrated Windows authentication to authenticate users and pass the information to the Web application. See *Integrated Windows Authentication* on page 201.
- Use an alternate login page and authenticate users against any of the other login options. See *Creating an alternate login page* on page 223.

Once a user is authenticated, the modules to which the user has access are defined by the back-end system. For example, if you are using AssetCenter and a user does not have access rights to a particular table in AssetCenter, the user cannot access the corresponding module in the Web application. 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 Get-Resources 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. Get-Resources automatically generates login modules for each Peregrine adapter installed on the system.

The default login module settings are:

- 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
<code><jaas_config></code> <code></jaas_config></code>	Define a custom JAAS configuration. All JAAS configuration settings must be between these two tags.
<code><jaasConfiguration></code> <code></jaasConfiguration></code>	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 <code>CustomConfig</code> , then you must use the tags <code><CustomConfig></code> and <code></CustomConfig></code> 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 Get-Resources assumes the name is the logical name of a non-OAA LoginModule.

Get-Resources 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 194 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 196 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: 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 Get-Resources 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

Module Name	Status	Scenario 1	Scenario 2	Scenario 3
LoginModule3	required	pass	pass	pass
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
<code>user.provider.url</code>	required	<p>Use this option to provide the URL to the starting point in your directory service where you want to search for users.</p> <p>For example, <code>ldap://server/dc=peregrine,dc=com</code></p> <p>Note: This option corresponds to the Java constant <code>Context.PROVIDER_URL</code>.</p>
<code>security.principal</code>	optional	<p>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, <code>uid=user,dc=peregrine,dc=com</code></p> <p>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.</p> <p>Note: This option corresponds to the Java constant <code>Context.SECURITY_PRINCIPAL</code>.</p>

Option	Use	Description
security.credentials	optional	<p>Use this option to define the password for the <code>security.principal</code> user. This option should only be used in conjunction with the <code>security.principal</code> option.</p> <hr/> <p>Important: If you are using a simple security authentication protocol, then this password may be passed as plain text.</p> <hr/> <p>Tip: To safeguard this password, either enable SSL (set the <code>security.protocol=ssl</code> option) or use an <code>security.authentication</code> that protects passwords.</p> <p>Note: This option corresponds to the Java constant <code>Context.SECURITY_CREDENTIALS</code>.</p>
security.protocol	optional	<p>Use this option to enable or disable an SSL connection between the <code>JndiLoginModule</code> and your directory server. This option has two possible values:</p> <ul style="list-style-type: none"> ■ <code>simple</code> (Default setting) ■ <code>ssl</code> <p>Note: This option corresponds to the Java constant <code>Context.SECURITY_PROTOCOL</code></p>
security.authentication	optional	<p>Use this option to enable or disable anonymous binding to your directory service. Typically, this option has one of two values:</p> <ul style="list-style-type: none"> ■ <code>none</code> (Default setting) ■ <code>simple</code> <p>Note: If you do not specify a value for <code>security.principal</code> then <code>security.authentication</code> defaults to a value of <code>none</code>. Likewise, if you set <code>security.authentication</code> to <code>simple</code> but <code>security.credentials</code> is omitted or has zero length, then <code>security.authentication</code> resets to <code>none</code>.</p> <p>Note: This option corresponds to the Java constant <code>Context.SECURITY_AUTHENTICATION</code>.</p>

Option	Use	Description
<code>user.search.scope</code>	optional	<p>Use this option to specify the number of levels to descend when searching for the user being authenticated by <code>user.provider.url</code>. This value must be an integer. The default value is 1.</p> <p>Note: This option corresponds to the Java constant <code>SearchControls.ONELEVEL_SCOPE</code>.</p>
<code>group.provider.url</code>	optional	<p>Use this option to provide the URL to the starting point in your directory service where you want to search for groups.</p> <p>For example, <code>ldap://server/dc=peregrine,dc=com</code></p> <p>Note: This option corresponds to the Java constant <code>Context.PROVIDER_URL</code>.</p>
<code>group.search.scope</code>	optional	<p>Use this option to specify the number of levels to descend when searching for a group. This option should only be used with <code>group.provider.url</code>. This value must be an integer. The default value is 1.</p> <p>Note: This option corresponds to the Java constant <code>SearchControls.ONELEVEL_SCOPE</code>.</p>
<code>group.search.objectClass</code>	optional	<p>Use this option to specify the name of the LDAP group objectClass. Valid values are:</p> <ul style="list-style-type: none"> ■ <code>groupOfNames</code> (Default value) ■ <code>groupOfUniqueNames</code>. ■ <code>groupOfUrls</code> <p>Note: Either <code>groupOfNames</code> or <code>groupOfUniqueNames</code> can be used to define static groups in LDAP, but they may not be used together.</p> <p>If you choose the <code>groupOfUrls</code> option, then you are configuring dynamic groups. No additional configuration settings are required to recognize dynamic groups.</p>
<code>storeIdentity=true</code>	optional	Use this option to store a reference to the User being authenticated.
<code>clearIdentity=true</code>	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.

```
<jaas_config>
  <jaasConfiguration>myConfig</jaasConfiguration>
    <myConfig>ldap</myConfig>

  <ldap>
    <loginModule>
      com.peregrine.oaa.security.JndiLoginModule
    </loginModule>
    <options>
      user.provider.url=ldap://server/dc=peregrine,dc=com;
      group.provider.url=ldap://server/dc=peregrine,dc=com
    </options>
  </ldap>
</jaas_config>
```

Standard Sun Microsystems JAAS configuration

The standard JAAS configuration option authenticates users against the Sun Microsystems 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.

Get-Resources uses Integrated Windows Authentication as follows:

- The user logs in to a Windows XP/2000/NT 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 circumstances are exceptions to the normal Integrated Windows Authentication login process:

- The Windows user has already registered with a back-end database adapter. When this occurs, Get-Resources asks the user to register and enter profile information. Get-Resources then lets the user log in and stores this information for future login attempts.
- The Windows user name is not already registered as an Administrator in the back-end system. When this occurs, the web application does not proceed with automatic login. The user is presented with another login screen and is asked to verify their password. 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 Get-Resources 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 a seven-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 203.
- Step 2** Install and configure Get-Resources with Apache and Tomcat. See *Preparing to configure Integrated Windows Authentication* on page 203.

Note: Tomcat/Apache is the preferred configuration for Get-Resources.
- Step 3** Set Web server properties for the `login.asp` file. See *Setting Web server properties for the login.asp file* on page 203.
- 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 205.
- Step 5** Set Web server properties for the `loginverify.asp` file. See *Setting Web server properties for the loginverify.asp file* on page 208.

- Step 6** Set the **Require Windows NT Challenge/Response Authentication** parameter, and optionally the **Default User Login Name** and **Default Login User Password** parameters from the Get-Resources administration page. See *Setting the Admin parameters* on page 209.
- Step 7** Optionally, define the **LogoutURL** from the Get-Resources administration page. This step is necessary when Get-Resources and IIS reside on different servers. See *Setting up the LogoutURL* on page 210.

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 AssetCenter or ServiceCenter administrator to verify that users have Operator records. Create an Operator record as needed.

Preparing to configure Integrated Windows Authentication

Note: If you are not using the preferred Tomcat/Apache configuration, skip this section.

- 1 Install and configure Get-Resources 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 Get-Resources 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 Get-Resources 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">
```

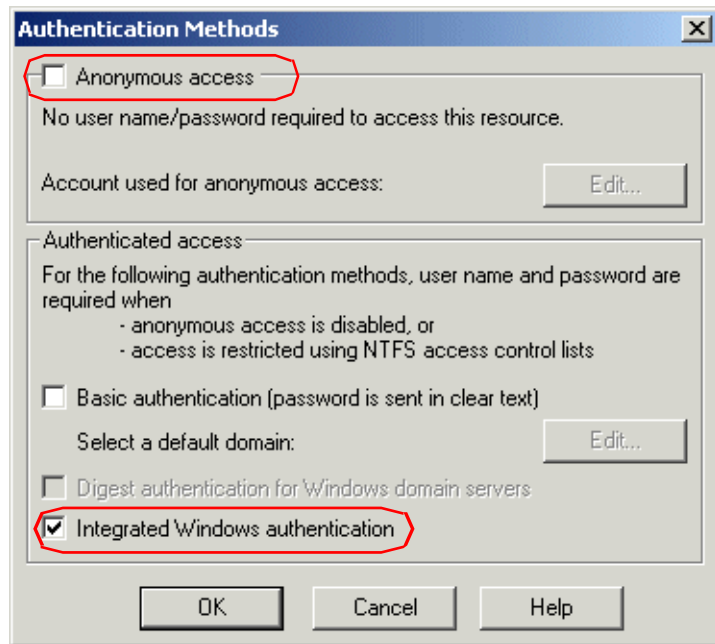
to:

```
<FORM name="f" action="
"http://<apacheserver.mycompany.com>/oaa/login.jsp" method="post">
```

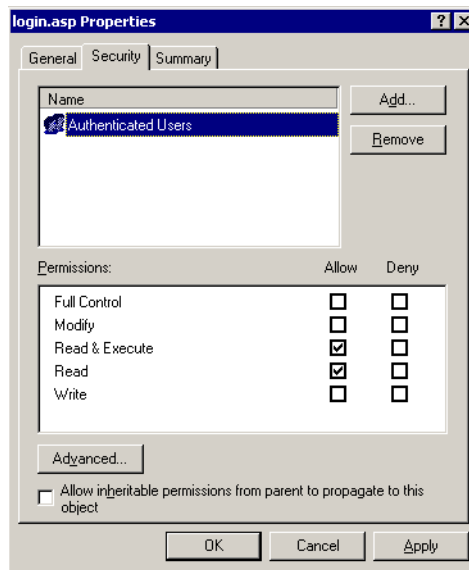
- 2 Open the IIS Management Console (**Start>Programs>Administrative Tools>Internet Information Services**).
- 3 Click on the oaa virtual directory.
- 4 Right-click on login.asp and select **Properties**.
- 5 Select the **File Security** tab.
- 6 Click **Edit** in the **Anonymous Access and Authentication 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



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">
```

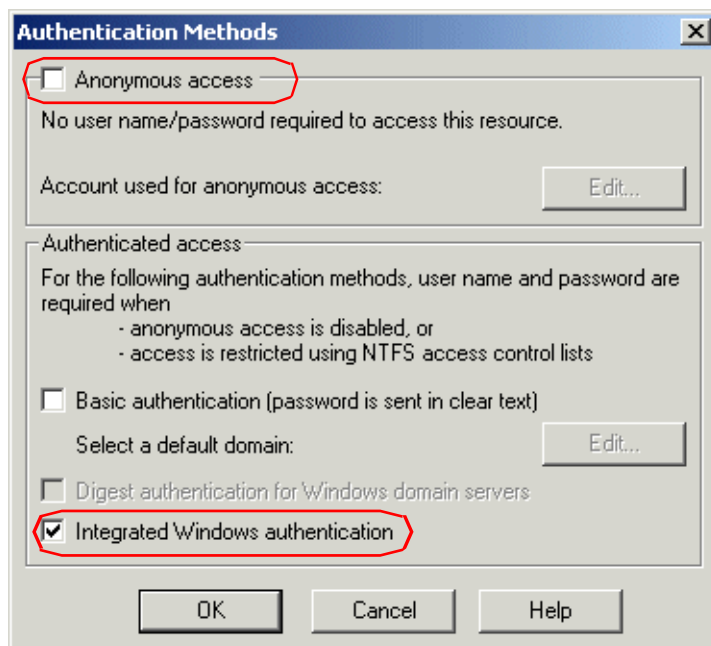
to:

```
<FORM name="f" action="http://<apacheserver.mycompany.com>/oaa/e_login_main_start.do" method="post">
```

- 2 Open the IIS Management Console (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.

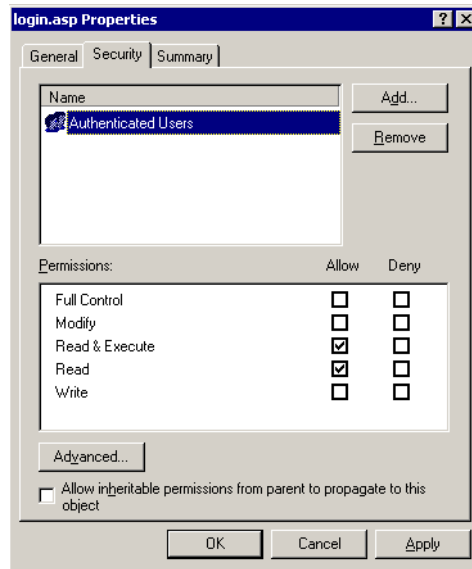
Clear the Anonymous access check box.

Select the Integrated Windows authentication check box.



- 7 Click **OK** on all windows displayed 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**

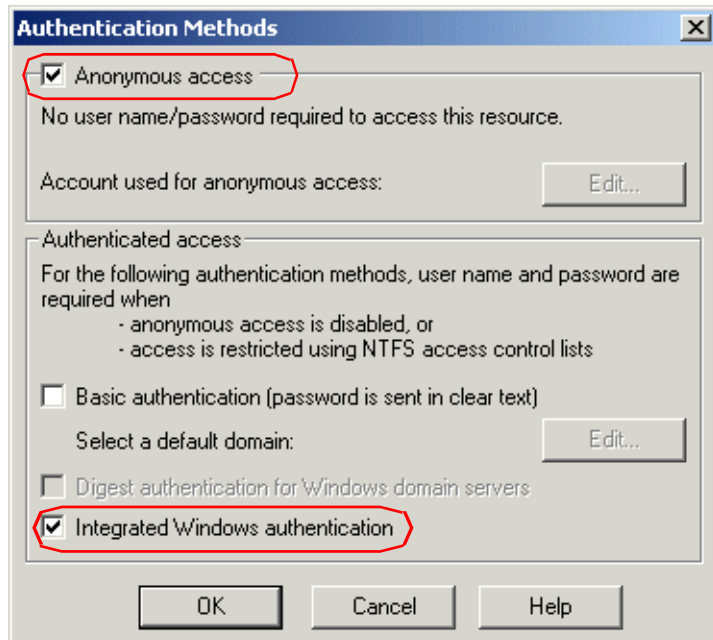
- Read – Allow



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** on all windows displayed until you return to the Microsoft Management Console.
- 8 Close the Management Console.

Setting the Admin parameters

You must set the **Require Windows NT Challenge/Response 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 Windows NT Challenge/Response Authentication:

- 1 Open a Web browser.
- 2 Enter the following URL: `http://<webserver>/<ooa>/admin.jsp` in the browser address field (where `<webserver>` is the name of your Web server and `<ooa>` 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**.

Select the **Yes** option in **Require Windows NT Challenge/Response Authentication** to allow only Windows users to log in.

The screenshot shows the 'Admin Settings' page with a sidebar on the left containing navigation links: Admin, Control Panel, Deployed Versions, Server Log, Settings (highlighted), Show Script Status, Show Message Queue, Show Queue Status, Import / Export, Adapter, Transactions/Minute, IBM Websphere Portal Integration. The main content area is divided into two columns. The left column contains several text input fields: Logout URL, Server URL (http://localhost/ooa/login.jsp?_bookmark=), Message URL Prefix, Help URL Prefix, Loginverify.asp URL prefix, and Default Login User Password. The right column contains several text input fields: Destination URL displayed when a user logs off of system, Link back to server, Defines the http domain used to build URL references with coded messages, Defines the URL Prefix used to access help forms, Enter URL prefix for loginverify.asp if IIS is on a different server than OAA, and Optional default user name. The 'Require Windows NT Challenge/Response Authentication' option is highlighted with a red circle and has the 'Yes' radio button selected. Below this option are the 'Default Login User Name' (Hartke) and 'Default Login User Password' fields.

- 5 From the **Common** tab, set the **Require Windows NT Challenge/Response 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 Server**.

Setting up the LogoutURL

Note: This step is necessary when Get-Resources and IIS reside on different servers.

- 1 From the Administration home page (see *To set Windows NT Challenge/Response Authentication*: on page 209), 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 Server**.

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>/<oaa>/login.asp** 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 Verify that access to Get-Resources is what you expected based on the settings you chose for the **login.asp** and **loginverify.asp** files.

Integrating with single sign-on tools

You can integrate Get-Resources with a single sign-on tool such as SiteMinder to eliminate displaying the Get-Resources login screen. When you integrate with a single sign-on tool, Get-Resources 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 Get-Resources 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 201.

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 Get-Resources. 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 Get-Resources 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:

```

Add JSP code here to          <%@ include file="jspheader.jsp" %>
obtain the user name         <%
of the person that the
single sign-on tool has
authenticated _____ // Add JSP code that obtains proper user name from
                           // the third party single-sign on tool
                           // ...

                           // Replace "user" with the user name obtained above
Replace the value _____ String sUser = "user";
"user" with the user
name obtained from
your single sign-on tool    // 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 Get-Resources from a single sign-on tool

You can use the following steps to test access to Get-Resources 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 Get-Resources home page.

Note: If you saved the automatic login page with a different file name, then use that file name instead of `autologin.jsp`.

Authentication models

The following sections discuss:

- *ServiceCenter authentication components*
- *OAA contact and operator associations*
- *Regular operator authentication*
- *Contact-based authentication*

ServiceCenter authentication components

There are two components of the ServiceCenter authentication model: the **Operator** file and the **Contacts** file.

The **Operator** file contains the following keys:

- The `name` field is the primary key (unique and indexed).
- The `full.name` field is a foreign key to the contact table. It represents the contact associated with the operator. It is indexed, it can be empty, and several operators can have the same value for this field. The value of the `full.name` field, when not empty, represents the value of the `contact.name` field in one of the records in the contacts file.

The **Contacts** file contains the following keys:

- The `contact.name` field is the primary key; it is unique and indexed.
- The `user.id` field is indexed and is a "no duplicate" field; it can be null, but must be unique if not null. When contact-based authentication is enabled, the `user.id` field is the key used to look up contacts.

OAA contact and operator associations

OAA approaches contact and operator handling by allowing ServiceCenter administrators to customize their **Contacts** and **Operator** files, and to use **Contacts** and **Operator** associations that differ from OAA defaults.

The OAA schemas allow flexibility in defining associations between records in the **Contacts** and **Operator** files. These OAA schemas provide a logical view that is “wrapped around” their physical implementations. OAA provides attribute names that correspond to each type of lookup operation. Therefore, for an administrator, customizing the lookup is as simple as creating a schema extension on the **Profile** or the **Contact** schema. For more information about schemas refer to the “Schemas” chapter in this guide.

Regular operator authentication

Name and password pairs are validated against the existing operator in the operator table. In addition, the presence of a contact that corresponds to the operator's contact is queried based on the fields mentioned below.

Note: If a contact for the corresponding operator's contact is not found, OAA creates a contact automatically.

Algorithm for looking up contacts

The Contact schema has the following attributes:

Logical name	Mapping in profile schema	Mapping in contact schema
OperatorContactKey1	full.name	contact.name
OperatorContactKey2	Name	user.id

Using these attributes, the lookup algorithm is the following:

- 1 Read the values for OperatorContactKey1 and OperatorContactKey2 in the Profile schema whose UserName equals the UserName (login) of the operator who is logged in.
- 2 Search the Contact schema for a record whose Id is the value of OperatorContactKey1.
- 3 If exactly one record is found, return this contact's Id.
- 4 If no record, or more than one record, is found, search the Contact schema for a record whose Id equals the value of OperatorContactKey2.
- 5 If exactly one record is found, return this contact's Id.
- 6 If no record, or more than one record, is found, return null and attempt to create the contact. (See the section *Contact creation* below.)

Contact creation

All the information from the Profile record for the logged in operator is used to create a record in the **Contact** schema. Therefore, all the Profile values that have a corresponding attribute in the **Contact** schema are saved in the database. In addition, the ProfileId record in the **Contact** schema (see below for mapping) is assigned the value of the Profile record's Id in order map the **Contact** to the Profile. The following tables describe both the logical and physical mappings of particular fields of interest during contact creation.

Logical mapping

Logical name in Profile schema	Logical name in Contact schema
Id	ProfileId
UserName	UserName
FullName	Id

Physical mapping

Logical name in Profile schema	Logical name in Contact schema
Name	operator.id
name	user.id
full.name	contact.name

Contact-based authentication

This section describes an alternate authentication scheme that automatically verifies Windows users as ServiceCenter contacts.

You can configure Get-Resources to automatically log in specific groups of authenticated Windows users as one or more pre-defined Operators in ServiceCenter. Each group of Windows users has its own login page.

Note: The authentication scheme described below requires that both the user who is logged into the machine running the client browser *and* the IIS server reside either in the same domain, or in different domains that have a trusted relationship.

Perform the following steps:

- Step 1** Look up the contact at login. See *Looking up the contact at login* on page 216.
- Step 2** Choose or create one Operator record in ServiceCenter for each group of Windows users you want to authenticate. See *Creating an Operator record in ServiceCenter* on page 217.
- Step 3** Create a contact record in ServiceCenter for each Windows user who you want to be able to log in. See *Creating a contact record* on page 217.
- Step 4** From the Windows domain server, add a Windows group for each Operator that you defined in step 2. Refer to your Windows documentation for more information on adding groups. See *Adding groups* on page 218.
- Step 5** Create a login ASP file for each Operator defined in step 2. See *Configuring the login ASP file* on page 218.
- Step 6** Configure each login ASP file to be exclusive to each Windows group defined in step 4. See *Setting properties for the login ASP file* on page 219.
- Step 7** On the Get-Resources Admin module Settings page, click the Common tab, scroll down to the Encoding, Locales, and Sessions section, and make sure that **Require Windows NT Challenge/Response Authentication** is set to **No**.
- Step 8** Edit `local.xml` in `<application server>\oaa\WEB-INF` to define the passwords for each Operator defined in step 4. See *Editing the local.xml file* on page 221.

Looking up the contact at login

OAA uses the ServiceCenter the `user.id` field in the contacts file to look up a contact for contact-based authentication. However, some administrators use this field to hold employee IDs (such as numeric employee IDs, badge number, and Social Security number) rather than network names (which are applicable when Integrated Windows Authentication is enabled). `UserName` is the logical name in the Contact schema for the `user.id` field. Through a schema extension, administrators can customize this to point to a different field or a newly defined field.

Logical name	Mapping in Contact schema
<code>UserName</code>	<code>user.id</code>

The contact lookup algorithm ensures that there is only one match for a given `UserName`. Otherwise, authentication is denied.

Creating an Operator record in ServiceCenter

Choose or create one Operator record for each group of users or role you want to access Get-Resources. Each Operator must have a password and a list of capability words. For example, you can define one Operator with default access (**scdefault**) and one Operator with manager access (**scmgr**). Refer to your ServiceCenter documentation for more information on adding Operator records.

The following procedures describe how to use **scdefault** and **scmgr** as the Operators.

Using Operator records in ServiceCenter:

- 1 Create two Operator records: **scdefault** and **scmgr**.
Refer to your ServiceCenter documentation for information on adding Operator records.
- 2 Add the Get-Resources capability words you want users assigned to this Operator to have. For example:

Operator	Capability words
scdefault	getit.service getit.personalization.default
scmgr	getit.service getit.itemployee getit.itmanager getit.personalization.default

Note: Each Operator will use its own login page.

In this example, users who log in to **logindefault.asp** have the capabilities of the **scdefault** Operator in ServiceCenter. Users who log in to **loginmgr.asp** have the capabilities of the **scmgr** Operator in ServiceCenter.

- 3 Assign a password to each Operator.

Note: The password must match the password defined in *Editing the local.xml file* on page 221.

Creating a contact record

Create a contact record for each Windows user who you want to be able to log in. The Employee ID field of the contact record must match the Windows user name exactly, including upper- and lower-case.

For more information about creating contact records, see the Service Center *System Administration Guide*.

Adding groups

You must have an equivalent Windows group for each Operator that you want to authenticate. For example:

Operator	Suggested group
scdefault	Authenticated Users (default Windows group)
scmgr	Managers (created on domain server)

Refer to your Windows documentation for adding groups to Windows.

Configuring the login ASP file

You must configure or create a separate login ASP file for each Operator you define (see *Creating an Operator record in ServiceCenter* on page 217). Each file needs a unique name.

Two sample login ASP files, `logindefault.asp` and `loginmgr.asp` are in the Get-Resources deployment directory: `<application server>\oaa`

To configure the login ASP file:

- 1 Create a unique login file for each Operator.

For example, create `logindefault.asp` for `scdefault` and create `loginmgr.asp` for `scmgr`.

- a Copy `logindefault.asp` from the deployment folder:
`<application server>\oaa`
- b Paste the file in the same folder and rename the copied file.

Note: Whatever file name you choose becomes part of the URL users enter to access Get-Resources. For example, if the file name is `mylogin.asp`, the URL is: `http://yourhostname/oaa/mylogin.asp`.

- 2 Edit the value of the OPERATOR form input to match the Operator you defined in *Creating an Operator record in ServiceCenter* on page 217.

```

...
<FORM name="f" action="login.jsp" method="post">
  <INPUT type="hidden" name="AUTH_TYPE" value="<%=sType%>" />
  <INPUT type="hidden" name="AUTH_USER" value="<%=sUser%>" />
  <INPUT type="hidden" name="AUTH_KEY" value="<%=sKey%>" />
  <INPUT type="hidden" name="OPERATOR" value="scdefault" />
</FORM>
...

```

The value of the OPERATOR must match the Operator name.

- 3 Save and close the file.

Setting properties for the login ASP file

You must configure each login ASP file to be exclusive to each Windows group. This requires changing the authentication method in IIS and setting the file security properties in Windows.

To change the authentication method in IIS:

- 1 Open the IIS Management Console (Start > Programs > Administrative Tools > Internet Information Services).
- 2 Navigate to the oaa virtual directory.
- 3 For each Operator, navigate to the ASP file that you created in *Configuring the login ASP file* on page 218.

For example, navigate to logindefault.asp for scdefault; navigate to loginmgr.asp for scmgr.

- 4 Right-click on the file 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.

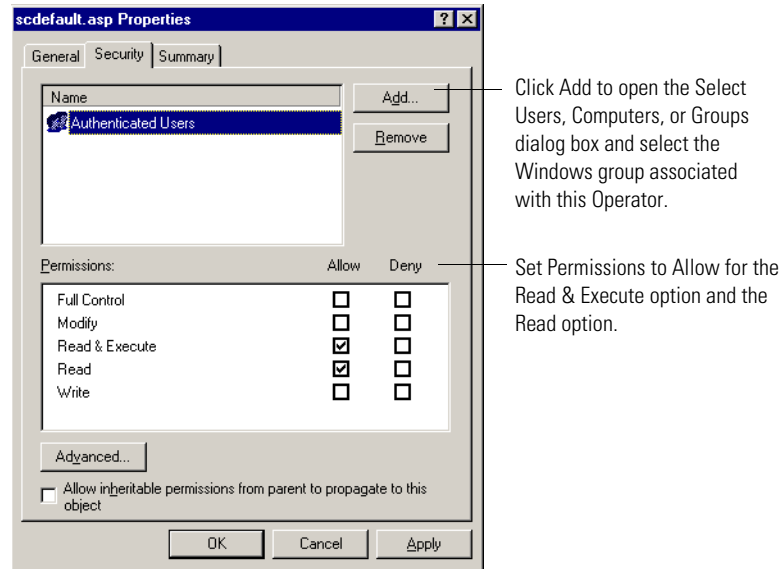


- 7 Click OK on all windows displayed until you return to the Microsoft Management Console.

To set the file security properties in Windows:

- 1 Open Windows Explorer.
- 2 Browse to your deployment folder: <application server>\oaa
- 3 Update the following login ASP properties.
 - a Right-click on your login ASP file; for example, scdefault.asp, and click **Properties**.

- b Add the user group associated with this Operator; for example, **Authenticated Users**.



- c Grant the following **Permissions** to the **Authenticated Users** group:

- **Read & Execute** – Allow
- **Read** – Allow

- d Click **OK**.

- 4 Repeat step 3 for each login ASP file.
- 5 On the Get-Resources Admin module Settings page, click the Common tab, scroll down to the Encoding, Locales, and Sessions section, and make sure that **Require Windows NT Challenge/Response Authentication** is set to **No**.

Editing the local.xml file

You must identify the password for each Operator that you defined in the local.xml file. This file is located at:

<application server>\oaa\WEB-INF\local.xml.

To edit the local.xml file:

- 1 Using a text editor, edit local.xml.

The default location is:

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

2 Add an XML entry for each Operator.

The tag has the format: `<[operator name]password>`

For example, for operators `scmgr` and `scdefault`, add the following inside the `<settings> ... </settings>` tags:

```
<scmgr>scmgr</scmgr>
<scmgrPassword>scmgr_password</scmgrPassword>
<scdefault>scdefault</scdefault>
<scdefaultPassword>scdefault_password</scdefaultPassword>
```

where `scmgr_password` is the ServiceCenter password assigned to operator `scmgr`, and `scdefault_password` is the ServiceCenter password assigned to operator `scdefault`.

Important: The password must match the Operator password in ServiceCenter.

3 Restart your application server for your changes to take effect.

AssetCenter authentication

Get-Resources 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 Get-Resources, but you will need an employee for your user. (See *Integrated Windows Authentication* on page 201.) 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, Get-Resources cannot authenticate the user directly through AssetCenter. The Get-Resources 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

Get-Resources 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 Get-Resources. The login name for Get-Resources 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 Get-Resources because Get-Resources 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 (JRun, 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 225.

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
<code>loginuser</code>	This is a required login parameter specifying the user name. You must specify a form input for this parameter.
<code>loginpass</code>	This is a required login parameter specifying the login password. You must specify a form input for this parameter.
<code>_locale</code>	This is an optional login parameter specifying the user's locale and regional display settings.
<code>_timezone</code>	This is an optional login parameter specifying the user's timezone.
<code>_theme</code>	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.

This chapter covers the following topics:

- *Browser issues* on page 227
- *Tomcat issues* on page 228
- *WebSphere Portal Server issues* on page 228
- *Sorting issues* on page 229

Browser issues

The following problems can result from the Internet browser you use to view Get-Resources.

Navigation Issue

When logged in to Get-Resources, using the browser Back, Forward, and Refresh buttons can cause unexpected behavior of Get-Resources forms.

Solution Do not use the browser navigation or Refresh buttons with Get-Resources forms displayed.

Issue When using the Microsoft Internet Explorer 5.5 browser, the following can occur:

- Icons fail to display in dataset 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).

Solution Upgrade to Internet Explorer 6.

Issue After changing a theme using the Change Themes page, clicking the Go Back button does not return you to the Home page.

Solution On the Activity menu in the sidebar, click My Home Page.

Issue 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.

Solution Create a new search to regenerate your list. Get-Resources does not cache what is on the screen.

Tomcat issues

The following problems involve issues with Tomcat as the application server.

Issue Tomcat fails to launch after a new version of the JDK is installed.

Solution When installing a new JDK, you must copy the JAR files from C:\Program Files\Peregrine\oaa\external (or to the installation location you specified) to the new JDK jre\lib\ext directory.

Issue Tomcat and Apache do not automatically start after a UNIX upgrade.

Solution Restart OAA by executing the command:
/usr/local/peregrine/bin/oaactl restart

WebSphere Portal Server issues

The following problems occur when using WebSphere Portal Server.

Issue The Web browser displays runtime errors when you view Get-Resources inside a WebSphere Portal Server page. This occurs with Internet Explorer version 5.50.4807.2300 SP2, but could also appear with other older browsers.

- Solution** Upgrade to the latest version of your Web browser.
- Issue** WebSphere Portal Server does not display the results of Get-Resources form in a new maximized window.
- Solution** To see form results in a maximized window, maximize the WebSphere portlet first, and then submit the form. The results display in the same portlet.
- Issue** If a user times out while in a maximized WebSphere Portal Server portlet, clicking on any link returns the user to `http://<server-name>/oaa/login.jsp` instead of the WebSphere Portal Server interface.
- Solution** Change the default time out parameter.
- Issue** There are various rendering errors when viewing Get-Resources portlets in WebSphere Portal Server when using Netscape 7.0 or Mozilla 1.0+. These errors are due to a known Mozilla bug. See Bugzilla Bug 67903 for additional details.
- Solution** Use a supported version of Internet Explorer to view WebSphere Portal Server portlets.

Sorting issues

- Issue** After you perform a search, you can sort displayed records by clicking a column heading once to sort in ascending order, and by clicking a second time to sort in descending order. However, when using the P4 database as the backend to ServiceCenter, you cannot sort displayed records by primary key values in descending order. For example, for Requests, the primary key field is Request number.
- Solution** Sort displayed records by primary key fields only in ascending order.

Index

A

- Activity menu 55
- adapter transactions, viewing 168
- Admin module
 - changing Settings 159
 - Control Panel 156
 - displaying message queues 166
 - generating web archive files 169
 - importing and exporting personalizations 168
 - message queues 166
 - script status 166
 - Server Log 165
 - Settings page 158
 - showing queue status 167
 - verifying script status 166
 - viewing adapter transactions 168
- Archway architecture
 - building blocks 17
 - clients 19
 - diagram 18
 - document manager 23
 - executing queries against a system 23
 - requests 21
 - XML 19
- AssetCenter 74
 - calculated field 178
 - catalog 33
 - Certification field 177
 - workflows 33

authentication

- AssetCenter 222
 - contact-based 215
 - models 213
 - overriding the login script 223
 - regular operator 214
 - users 191
- Automatic PO Generation workflow, disabling 175

B

- book
 - audience 10
 - organization 12

C

- calculated field 178
- capability words 183
- catalog, AssetCenter 33
- Certification field in AssetCenter 177
- changing passwords 173
- changing the Peregrine Portal layout 60
- changing themes 62
- collections
 - configuring 73
- components
 - adding Portal 57
 - creating new 55
- Control Panel 156
- conventions, typographical 11
- CSS files, editing 41

customer support 13

D

deploying themes 38
document manager 23
document schema definitions. See schemas
documentation, related 10

E

exporting personalized pages 82, 168

F

field size 80
field span 80
fields
 configuring 70
form details 171
form details, displaying 171
Form Info, displaying 169
form information, displaying 63
framesets, changing 46

G

getit.admin 185
getit.admin user rights 154
Get-Services
 overview 9

H

header graphic, changing 39

I

IBM Websphere portal 169
icons, personalizing 68
importing personalized pages 82, 168
Info button 171
Integrated Windows Authentication
 configuring 201
 security 182
ISO character encoding. See character encoding

J

JAAS
 authentication 191
 login modules 192

L

labels, personalizing 79
language
 login 54, 159
layers, changing 43
layout, changing
 MSIE 60
 Netscape Navigator 61
LDAP 182
Lightweight Directory Access Protocol 182
loadscript
 editing in schema subclasses 108
local.xml file 154, 159
log, form details 171
Logging 160
 file format 161
 file rollover 164
logging user sessions 173
login authentication 191
login language 54, 159
login modules, JAAS 192
login script, overriding 223
login.asp 210

M

message queues 166
message queues, displaying 166
monitoring user sessions 173
moving personalized pages 82, 168

O

overriding the login script 223

P

package.xml 107
parameters
 ServiceCenter securepassword 186
parameters, defining 159
password security 186
password, changing 173
passwords
 protecting 182
Peregrine Portal
 adding components 57
 personalizing 56

- Peregrine Portal, tailoring 37
- Peregrine Systems customer support 13
- Personalization
 - interface, described 67, 71
 - list of standard forms 66
 - settings 76
- personalization
 - adding fields 77
 - arranging field order 78
 - interface description 67
 - requirements 74
 - settings 75
 - user rights 76
- personalizations
 - adding 68
 - removing 68
- personalized pages
 - moving 82, 168
- personalizing
 - adding a new section to the field layout 78
 - changing field layout 78
 - field size 80
 - field span 80
 - forms 67–81
 - icons 68
 - labels 79
 - portal 56–63
 - read-only field 79
 - required field 80
- personalizing the Peregrine Portal 56
- PO generation, disabling workflow 175
- Portal components
 - Business View Authoring 81
 - making schemas visible to 81
- Portal Components, creating new 55
- preXSL, form details 171
- public schemas
 - making schema subclasses visible 107

Q

- queue status, displaying 167

R

- read-only field 79
- related documentation 10

- required field 80
- resetting the server 156

S

- scalability
 - OAA 19
- schema elements
 - 129
- schemas
 - defined 86
 - elements 118–137
 - extension folders 91
 - extensions 89–105
 - identifying schema used 90
 - locating 91
 - sample 87
 - subclassing 106
 - testing from a URL 22
 - uses for extensions 93
- script input, form details 171
- script output, form details 171
- script status 166
- script status, verifying 166
- scripts
 - testing from a URL 21
- Secure Sockets Layer 182
- securepassword parameter 186
- security
 - alternate login authentication 223
 - user authentication 191
 - Windows Integrated Authentication 201
- self-registration 172
- server log 165
- ServiceCenter 74
- Settings page 159
- SSL 182
- string files
 - translating 49, 50
- subdocuments
 - Configuring 71

T

- tailoring themes 37
 - changing framesets 46
 - changing layers 43

- changing stylesheets 41
- changing the header graphic 39
- deploying themes 38
- technical support 13
- terminology 11
- themes
 - deploying 38
 - tailoring 37
- themes, changing 62
- themes, creating 41
- translating strings 48
- typographical conventions 11

U

URL

- querying scripts and schemas from 21
- user registration 172
- user rights
 - getit.admin 154
- user rights, Personalization 75
- user session 171
- user sessions, logging 173
- user.log file 173

W

- web archive (war) files 169
- Websphere portal 169
- workflow, disabling 175
- workflows
 - Automatic PO Generation 35
 - Bundle Ordering 33
 - Request Approval 34
 - Request Status 36
 - Routing Request 34

