

Oracle® Retail Workspace

Implementation Guide

Release 13.1

June 2009

Copyright © 2009, Oracle and/or its affiliates. All rights reserved.

Primary Author: Nathan Young

Contributing Author: Anirudha Accanoor

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server - Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.

(ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun Microsystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Preface	xiii
Audience	xiii
Related Documents	xiii
Customer Support	xiii
Review Patch Documentation	xiv
Oracle Retail Documentation on the Oracle Technology Network	xiv
Conventions	xiv
Open-Source Third Party Applications for Oracle Retail Workspace	xiv
1 Introduction	
What is Oracle Retail Workspace?	1-1
2 Backend System Administration and Configuration	
Configuration Files	2-1
retail-workspace-page-config.xml	2-1
Top Level Element	2-2
Definition Elements	2-3
Subordinate Elements	2-7
ldap-config.xml	2-7
Logging	2-10
Jakarta Commons Logging	2-10
Log4J Logging	2-10
OC4J Logging	2-11
Internationalization	2-11
Translation	2-11
Set the Client to the Applicable Locale	2-12
Adding a New Resource Bundle to Retail Workspace	2-12
Deployment of New Resource Bundle(s)	2-13
Maintenance	2-13
Exporting and Importing Portlet Customizations	2-13
3 Technical Architecture	
Oracle Retail Workspace Technical Architecture Diagram and Descriptions	3-1
Oracle Retail Workspace Architecture	3-2
Retail Workspace User Interface Framework	3-3

Single Sign-on Overview and Topology	3-4
What is Single Sign-On?	3-4
What Do I Need for Oracle Single Sign-On?	3-4
Can Oracle Single Sign-On Work with Other SSO Implementations?	3-4
Can Oracle Retail Workspace Work with Other SSO Implementations?	3-5
Oracle Single Sign-on Terms and Definitions	3-5
Authentication	3-5
Dynamically Protected URLs	3-5
Identity Management Infrastructure.....	3-5
MOD_OSSO	3-5
Oracle Internet Directory	3-5
Partner Application	3-5
External Application.....	3-6
Realm	3-6
Statically Protected URLs.....	3-6
What Single Sign-On is not.....	3-6
How Oracle Single Sign-On Works	3-6
Statically Protected URLs.....	3-6
Dynamically Protected URLs	3-7
OSSO Installation Overview.....	3-8
Infrastructure Installation and Configuration	3-8
OID User Data	3-8
OID with Multiple Realms	3-9
User Management	3-9
OID DAS.....	3-9
LDIF Scripts	3-9
User Data Synchronization.....	3-9
External Application Integration	3-10

4 External Applications

Oracle Single Sign-On and External Applications	4-1
OSSO Prerequisites	4-2
Supported Types of Authentication	4-2
Security Considerations Launching External Applications	4-2
Defining an External Application in OSSO	4-3
Required Privileges.....	4-3
Launching OSSO External Application Management	4-3
Determining the Authentication Type	4-6
Determining Field Names and Values	4-6
Alternate Methods Determining OSSO External Application information	4-8
Testing an External Application Definition	4-8
Obtaining the OSSO External Application ID	4-8
Managing User Credentials	4-9
Configuring Oracle Retail Workspace to Launch External Applications	4-10
Retail Workspace Sample External Application JavaScripts	4-11
Installation.....	4-12
appinfo.xml	4-12

basicAppRemote.html	4-13
jSecAppORW.html	4-15
jSecAppRemote.html	4-17
j_security_check Applications	4-19
Oracle Retail Central Office	4-20
Defining the Central Office External Application	4-20
Obtain the OSSO Application ID	4-21
Installing the jSecAppORW.html or jSecAppRemote.html and Associated JavaScript Files	4-21
Edit appinfo.xml.....	4-21
Edit retail-workspace-page-config.xml.....	4-22
Central Office Launch Link	4-22
Central Office User Credentials Link.....	4-23
Launching Central Office	4-24
5 Security	
Overview	5-1
Deployment Descriptor Security Considerations	5-2
Single Sign-On.....	5-3
Roles and Groups	5-3
Permission Grants	5-3
Managing Permission Grants via the Permissions Management Tool.....	5-4
LDAP Access	5-5
Password Storage.....	5-5
External Servers and Web Services	5-6
6 Integration Methods and Communication Flow	
Overview	6-1
System to System Integration	6-1
Integration Interface Diagram.....	6-2
ORW and Oracle Retail Applications	6-2
Reporting Tools in ORW.....	6-2
Oracle Retail Workspace and Single Sign-on	6-3
Integrating with Oracle Retail Applications.....	6-3
Integrating with Report Servers	6-3
Exposing Reports Links in the Navigation Panel.....	6-3
The Webservices Login ID.....	6-4
The Webservices URL Prefix	6-4
Showing Reports in Dashboards.....	6-5
Integrating with Oracle BI Alerts.....	6-6
Exposing Oracle BI Alerts Link in the Navigation Panel	6-6
Displaying Oracle BI Alerts in an Alerts Portlet on a Dashboard.....	6-7
Integrating with BPEL Workflows and BPM Worklists	6-8
Configuring Role Based Access to BPEL Workflows	6-10

7 Functional Design and Overview

Product Overview	7-1
Single Sign-On	7-1
Central Launch	7-2
Role Based Security.....	7-2
Dashboard Creation and Viewing.....	7-2
Reports.....	7-2
Oracle BI Delivers Alerts.....	7-2
BPM Worklists	7-3
BPEL Workflows	7-3
User Management	7-3
Client Specific Customization	7-3
Example Roles and Dashboards.....	7-3

8 Customization Guide

Customizing Oracle Retail Workspace Navigation	8-1
Changing the ORW Configuration File	8-2
Internationalizing String Values in the Navigation Panel	8-3
Example: Creating and configuring a resource bundle.....	8-3
Securing Work Elements.....	8-4
Configuring Permission Grants When Using OID	8-5
Configuring Permissions Grants to Allow Portlet Customization and Personalization	8-7
Changing Existing Content in the Navigation Panel	8-9
Changing a Work Item's URL and Query String Parameters.....	8-9
Example: Changing the URL and parameters of a dashboard.....	8-9
Example: Changing the URL and parameters of an Oracle Retail application.....	8-11
Changing a Work Item's Label.....	8-11
Example: Changing the label of a work item.....	8-12
Example: Changing/setting the tooltip of a work item	8-12
Changing a Work List Title.....	8-13
Hiding a Work List	8-13
Example: Changing the "rendered" attribute to hide a work list.....	8-14
Making a Hidden Work List Visible.....	8-14
Hiding a Work Item.....	8-14
Example: Changing the "rendered" attribute to hide a work item	8-14
Making a Hidden Work Item Visible	8-15
Changing an Unsecure Work List to a Secure Work List.....	8-15
Example: Changing an unsecure work list to a secure work list	8-15
Changing an Unsecure Work Item to a Secure Work Item.....	8-15
Example: Changing an unsecure work item to a secure work item	8-16
Changing the ID of a Work Item or Work List	8-16
Example: Changing a Dashboard's ID	8-17
Customizing Reports Work Items	8-17
Oracle BI EE Work Item Configuration Options	8-20
BI Publisher Work Item Configuration Options.....	8-21
Adding New Content to the Navigation Panel	8-23
Adding Work Lists to the Navigation Panel.....	8-23

Changing Work List Folders	8-24
Adding Work Items to the Navigation Panel	8-24
Defining URL Query String Parameters for a Work Item	8-26
Adding Work Items that Launch a URL in the Content Area	8-26
Adding Work Items that Launch a URL in a Browser Window	8-27
Adding a Secure Dashboard to the Dashboards List	8-27
Removing Content from the Navigation Panel	8-28
Removing Work Items	8-28
Example: Removing an unsecure work item	8-28
Example: Removing an Oracle Retail application from the Applications work list	8-29
Example: Removing a Dashboard	8-29
Removing Work Lists	8-31
Example: Removing an Unsecure Work List that has Secure Work Item Descendents	8-31
Configuring the Workspace Home Page	8-33
Adding a Home Work Item	8-33
Deleting a Home Work Item	8-34
Editing a Home Work Item	8-34
Developing and Using Custom Configuration Elements	8-34
ORW Configuration Framework Design Overview	8-34
Steps for Developing and Using Custom Configuration Elements - Overview	8-35
Developing Custom Work Items	8-35
Example: Developing a custom work item	8-35
Customizing the ORW Branding and Look and Feel	8-43
Changing the ORW Logo and Application Name	8-43
Changing the ORW Logo	8-43
Changing the Application Name	8-43
Creating a New ORW Skin	8-44
Customizing the Simple Skin	8-44
Customizing the Default Skin	8-46
Style Selectors Explained	8-48
SkinsBean Explained	8-48
Additional Skin Information and References	8-49

9 Dashboard Development Tutorial

Introduction	9-1
Audience	9-1
Getting Started	9-1
Downloading Oracle JDeveloper 10.1.3.4 with the Web Center Extensions	9-1
Creating the Dashboard Application and Registering the Portlet Producer	9-2
Create a New JDeveloper Application	9-2
Verify Access to the Deployed Retail Workspace Portlet Producer	9-3
Register Portlet Producer	9-3
Planning your Dashboard	9-5
Dashboard Layout	9-6
Creating the Dashboard	9-6
Create the Dashboard Page (MyDashboard.jspx)	9-7
Layout the Dashboard	9-9

Set PanelCustomizable Properties	9-10
Add Portlets to the Dashboard.....	9-10
Set Portlet Parameters	9-14
Summary of Portlet Parameters	9-19
More Information about Page Definition Files.....	9-22
Adding Oracle Retail Skins to your Dashboard	9-26
What is a Skin?.....	9-26
Add the Oracle Retail Skin to your Dashboard	9-26
Troubleshooting	9-29
Adding the <i>retailPortalContentBody</i> Style Class to Dashboard Body	9-29
Securing the Dashboard.....	9-31
Run the ADF Security Wizard.....	9-32
Edit the web.xml Deployment Descriptor.....	9-37
Create the orion-web.xml Deployment Descriptor.....	9-38
Create/Update the adf-config.xml File	9-39
Edit the system-jazn-data.xml File and Add a Permission Grant.....	9-42
Adding Customization and Personalization.....	9-45
Deploying the Dashboard	9-53
Creating a Deployment Profile.....	9-53
Define a Connection to the PreConfigured OC4J.....	9-55
Edit the Preconfigured OC4Js system-jazn-data.xml.....	9-56
Deploy Dashboard to PreConfigured OC4J.....	9-56
Deploy the Dashboard to Generic Ear	9-57
Convert the Generic Ear to the Targeted Ear	9-58
Deploy the Targeted EAR to OAS	9-59
Deploying a Dashboard with Oracle Single Sign-On	9-59
Using the OIM Security Provider	9-59
Changing the Security Provider	9-59
Adding Users, Roles, and Permissions to OID LDAP.....	9-62
Adding your Dashboard to ORW.....	9-62
Create Permission Grants via the Permissions Management Administration Tool.....	9-64
Passing Parameters to a Dashboard.....	9-68
Convert Portlet Page Definition Parameter Variables	9-68
Convert Portlet text Attribute Value	9-71
References.....	9-72
Internationalizing a Dashboard	9-73
Configure the Supported Locales for the Dashboard Application	9-73
Internationalizing your Dashboard Application.....	9-74
Create a Resource Bundle for the Dashboard Application.....	9-74
Internationalize the Dashboard Page.....	9-75
Internationalizing your Dashboard within ORW	9-76
Create a Dashboard Resource Bundle for Use in the ORW Configuration.....	9-76
Modify the ORW Configuration File to Reference Strings in the Resource Bundles.....	9-78
Summary of Configuration Changes	9-80
Team Development Considerations	9-80

10 Troubleshooting

Logging.....	10-1
Caching.....	10-1
Troubleshooting.....	10-1

Preface

The Oracle Retail Workspace Implementation Guide provides detailed information useful for implementing the application. It helps you view and understand the behind-the-scenes processing of the application. In addition, this implementation guide includes information about customizing Oracle Retail Workspace and a tutorial on developing dashboards.

Audience

The Implementation Guide is intended for Oracle Retail Workspace application integrators and implementation staff.

Related Documents

For more information, see the following documents in the Oracle Retail Workspace 13.1 documentation set:

- *Oracle Retail Workspace Release Notes*
- *Oracle Retail Workspace Installation Guide*
- *Oracle Retail Workspace Administration Guide*
- *Oracle Retail Workspace Online Help*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

- <https://metalink.oracle.com>

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to recreate
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.1) or a later patch release (for example, 13.1.1). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code:

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Open-Source Third Party Applications for Oracle Retail Workspace

Software Provider: Apache Software Foundation

Software Name: Apache Commons

Software Version: 1.0.4

Jar File Name: commons-logging.jar

Provider Web Site: <http://jakarta.apache.org/commons/>

Software Provider: Apache Software Foundation

Software Name: Apache log4j

Software Version: 1.2.13

Jar File Name: log4j-1.2.13.jar

Provider Web Site: <http://logging.apache.org/log4j>

Software Provider: Apache Software Foundation
Software Name: Jakarta-ORO
Software Version: 2.0.5
Jar File Name: jakarta-oro-2.0.5.jar
Provider Web Site: <http://jakarta.apache.org/oro/>

Software Provider: Apache Software Foundation
Software Name: Jakarta Regexp
Software Version: 1.1
Jar File Name: jakarta-regexp-1.1.jar
Provider Web Site: <http://jakarta.apache.org/regexp/>

Software Provider: Apache Software Foundation
Software Name: Apache Xerces
Software Version: J_1.4.0-xml
Jar File Name: xerces-J_1.4.0-xml.jar
Provider Web Site: <http://xerces.apache.org/>

Software Provider: Intalio Inc., and others
Software Name: Castor XML code generator
Software Version: 1.1.1
Jar File Name: castor-1.1.1-codegen.jar
Provider Web Site: <http://www.castor.org/>

Software Provider: Intalio Inc., and others
Software Name: Castor XML data binding framework
Software Version: 1.1.1
Jar File Name: castor-1.1.1-xml.jar
Provider Web Site: <http://www.castor.org/>

Software Provider: Sun Microsystems
Software Name: Java Mail
Software Version: 1.4
Jar File Name: mail.jar
Provider Web Site: <http://java.sun.com/products/javamail/>

Software Provider: jQuery

Software Name: jQuery
Software Version: 1.2.6
Jar File Name: jquery.jar
Provider Web Site: <http://jquery.com/>

Software Provider: jQuery jFeed plugin
Software Name: jFeed
Software Version: 1.0
Jar File Name: jfeed.jar
Provider Web Site: <http://plugins.jquery.com/project/jFeed>

Software Provider: jQuery Cycle plugin
Software Name: Cycle
Software Version: 2.26
Provider Web Site: <http://plugins.jquery.com/project/cycle>

Software Provider: jQuery bgiframe plugin
Software Name: BGIFRAME
Software Version: 2.1.1
Provider Web Site: <http://plugins.jquery.com/project/bgiframe>

Introduction

This implementation guide serves as a reference to explain back end processes for Oracle Retail Workspace.

What is Oracle Retail Workspace?

Oracle Retail Workspace (ORW) provides a single point of access to Oracle Retail applications used by your business. It gives a business an integrated platform from which to access operational and analytical information through dashboards and reports from both internal and external sources.

The ORW application provides the following features:

- **Dashboards**—The ORW application provides the ability to integrate custom dashboards displaying a snapshot of one's business. The sample dashboards included with ORW illustrate some of the technology and portlets that are available.
- **Launch Pad**—The ORW application serves as a launch pad for your Oracle Retail applications and reports, allowing access to them from a single point, without the need to log in many times. Other Oracle applications can also be launched from ORW.
- **Reports**—The ORW application is integrated with the Oracle Business Intelligence Enterprise Edition (BI EE) and Business Intelligence Publisher (BIP) reports servers.
- **Oracle Internet Directory Delegated Administrative Services**—Many of the user and administrative tools in the ORW navigation panel are links to the Oracle Internet Directory (OID) Delegated Administrative Services (DAS) application.
- **Alerts**—The ORW application is integrated with Oracle BI Delivers Alerts. The ORW BI Delivers Alerts portlet can be launched from the navigation pane and can be displayed on a dashboard.
- **BPM Worklists**—The ORW application is integrated with BPM Worklists. The BPM Worklists application can be launched through the navigation pane and the ORW Worklists portlet can be displayed on a dashboard.
- **BPEL Workflows**—The ORW application is integrated with BPEL Workflows. The BPEL Workflow services can be launched through the navigation pane and the ORW Workflows portlet can be displayed on a dashboard.
- **BPEL Administration**—The ORW application is integrated with the BPEL Admin console. The BPEL admin console can be launched from the ORW navigation pane.

Backend System Administration and Configuration

This chapter of the implementation guide is intended for administrators who support and monitor the running system. The content in this chapter is not procedural. It is meant to provide descriptive overviews of the key system parameters.

Configuration Files

When retailers install ORW into their environment, they must update the values for certain key system configuration parameters to their specific settings. These system configuration parameters are described in this section. Parameters that do not require modification are not included in this section.

There are two configuration files that clients normally edit:

- [retail-workspace-page-config.xml](#)
- [ldap-config.xml](#)

These files are found in the following directory:

```
$ORACLE_HOME/j2ee/<oc4j instance>/RetailWorkspace
```

<oc4j instance> is the name of the OC4J instance where ORW is installed.

retail-workspace-page-config.xml

The `retail-workspace-page-config.xml` file contains the branding configuration, the work lists, work items, secure work items, and default home pages for ORW.

The ORW installer prompts for the retail application URLs and parameters to populate the file. If the Oracle Retail applications have been installed prior to ORW, their URLs and parameters can be entered at ORW's install time. However, if a retailer installs a new application after ORW is installed, the `retail-workspace-page-config.xml` file needs to be edited to reflect the new application.

The file, as supplied in the release, is an example configuration and specifies demonstration dashboards and an artificial classification of applications into functional areas. It is very likely that retailers will customize this file after initial installation.

In the following example, the entry for Oracle Retail Merchandising System (RMS) consists of the main URL string plus one parameter named **config**. The installer prompts for these values and inserts them into the property files. The installer

property files hold the values for the `deploy.retail.product.rms.url` and `deploy.retail.product.rms.config` properties.

```
<url>@deploy.retail.product.rms.url</url>
<parameters>
  <parameter name="config">
    <value>@deploy.retail.product.rms.config@</value>
  </parameter>
</parameters>
```

Suppose RMS was installed on `mycomputer.mycompany.com`, port `7777`, using a standard installation and configured with the application name of **rmsprod**. To access RMS directly from a browser, one would enter the following URL:

```
http://mycomputer.mycompany.com:7777/forms/frmservlet?config=rmsprod
```

After installation, the values used to access the application in the `retail-workspace-page-config.xml` file would be similar to the following:

```
<url>http://mycomputer.mycompany.com:7777/forms/frmservlet</url>
<parameters>
  <parameter name="config">
    <value>rmsprod</value>
  </parameter>
</parameters>
```

ORW is configured with a default `retail-workspace-page-config.xml` file. This file has a simple XML structure that relies on the JSF expression language (EL) for flexibility. Configurable properties may be defined inline with static, constant values. Several of the properties may also be configured using JSF EL expressions that reference resource bundle properties or JSF managed beans. EL expression evaluation occurs on the application server during the JSF request lifecycle for the Retail Workspace JSP page.

The following elements are in alphabetical order, not in order of appearance.

Top Level Element

retail-portal-page

The `<retail-portal-page>` element is the root of the configuration information hierarchy, and contains nested elements for all of the other configuration settings. The `<retail-portal-page>` has the following attributes:

- `application-title`: specifies the application's title in the browser. This defaults to Oracle Retail Workspace if not specified. (optional)
- `branding-app-text`: specifies the application name in the header. Defaults to Retail Workspace if not specified. (optional)
- `branding-alt-text`: specifies the 'alt' text for the corporate branding image. Defaults to Oracle if not specified. (optional)

retail-portal-page Child Elements

- [branding-uri](#)
- [home](#)
- [navigation-lists](#)
- [resource-bundles](#)

Definition Elements

biee-reports-work-item

The <biee-reports-work-item> element specifies a work item for BIEE reports. The <biee-reports-work-item> has the following attributes:

- **id**: specifies the unique ID for this object (required)
- **display-string**: specifies the display string for this element (required)
- **rendered**: specifies if this element is to be rendered or not (required)
- **tooltip**: specifies the tooltip string. If not specified, defaults to value of display-string attribute. (optional)
- **launchable**: specifies if this element is launchable or not. Defaults to false. (optional)

biee-reports-work-item Child Elements

- [custom-attributes](#)
- [parameters](#)

bip-reports-work-item

The <bip-reports-work-item> element specifies a work item for BIP reports. The <bip-reports-work-item> has the following attributes:

- **id**: specifies the unique ID for this object (required)
- **display-string**: specifies the display string for this element (required)
- **rendered**: specifies if this element is to be rendered or not (required)
- **tooltip**: specifies the tooltip string (optional)
- **launchable**: specifies if this element is launchable or not (optional)

bip-reports-work-item Child Elements

- [custom-attributes](#)
- [parameters](#)

child-work-items

The <child-work-item> element specifies the child <work-item> and/or <secure-work-item> elements of a parent <work-item> or <secure-work-item>.

Work items and work lists are hierarchical. The <child-work-item > element holds the children of the current work item. This is also an optional element.

child-work-items Child Elements

- [work-item](#)
- [secure-work-item](#)

custom-attribute

The <custom-attribute> specifies a custom attribute name-value pair. The <custom-attribute> has the following attribute:

- **name**: specifies a string that is the name of the custom attribute (required)

custom-attribute Child Element

- [value](#)

custom-attributes

The <custom-attributes> element specifies one or more <custom-attribute> elements (optional)

custom-attributes Child Element

- [custom-attribute](#)

homes

The <homes> element specifies the home elements for ORW. This element can contain multiple <home> elements.

homes Child Element

- [home](#)

navigation-lists

The <navigation-lists> element currently supports only one <navigation-list>. It is a child element of the <retail-portal-page> element.

navigation-lists Child Element

- [navigation-list](#)

navigation-list

ORW supports only one navigation list child element for the <navigation-lists> element. The <navigation-list> has the following attribute:

- **id**: the unique ID for the navigation list (required)

navigation-list Child Element

- [work-lists](#)

parameters

The <parameters> element specifies one or more <parameter> elements.

parameters Child Element

- [parameter](#)

parameter

The <parameter> element specifies a query string parameter name-value pair. The <parameter> has the following attribute:

- **name**: specifies a string that is the name of the URL query string parameter (required)

Note: ORW supports only ASCII characters in the parameter "name" attribute.

parameter Child Element

- [value](#)

resource-bundles

The <resource-bundles> element specifies the resource bundles for ORW. It is a child element of the <retail-portal-page> element. This element can contain multiple <resource-bundle> elements.

resource-bundles Child Element

- [resource-bundle](#)

secure-work-item

The <secure-work-item> element is a child element of a work list's or secure work list's <work-items> element. The <secure-work-item> element may also be a child element of a parent work item's <child-work-items> element. The <secure-work-item> has the attributes:

- **id**: specifies the unique ID for this object (required)
- **display-string**: specifies the display string for this element (required)
- **rendered**: specifies if this element is to be rendered or not (required)
- **tooltip**: specifies the tooltip string (optional)
- **launchable**: specifies if this element is launchable or not. Defaults to false. (optional)
- **show-in-content-area**: boolean value to specify if the launched URL should show in the content area. Defaults to false (optional)
- **target-frame**: specifies a string that indicates the target frame. If not specified, defaults to "_blank" if show-in-content-area is false. To show content in the content area, specify target-frame as "_iframe". (optional)

secure-work-item Child Elements

- [child-work-items](#)
- [url](#)
- [parameters](#)

secure-work-list

The <secure-work-list> element contains one <work-items> element which may contain one or more <work-item> and/or <secure-work-item> elements. The <secure-work-list> has the following attributes:

- **id**: specifies the unique ID for this object (required)
- **display-string**: specifies the display string for this element (required)
- **tooltip**: specifies the tooltip string (optional)
- **rendered**: specifies if the work list is to be rendered or not (required)

secure-work-list Child Elements

- [icon-uri](#)
- [work-items](#)

work-item

The <work-item> element is a child element of a work list's or secure work list's <work-items> element. The <work-item> element may also be a child element of a parent work item's <child-work-items> element. The <work-item> has the following attributes:

- `id`: specifies the unique ID for this object (required)
- `display-string`: specifies the display string for this element (required)
- `rendered`: specifies if this element is to be rendered or not (required)
- `tooltip`: specifies the tooltip string. If not specified, defaults to value of `display-string` attribute. (optional)
- `launchable`: specifies if this element is launchable or not. Defaults to false. (optional)
- `show-in-content-area`: boolean value to specify if the launched URL should show in the content area. Defaults to false. (optional)
- `target-frame`: specifies a string that indicates the target frame. If not specified, defaults to "_blank" if `show-in-content-area` is false. To show content in the content area, specify `target-frame` as "_iframe". (optional)

work-item Child Elements

- [child-work-items](#)
- [url](#)
- [parameters](#)

work-items

The <work-items> element specifies one or more <work-item> or <secure-work-item> elements. It is a child element of <work-list> or <secure-work-list>.

work-items Child Elements

- [work-item](#)
- [secure-work-item](#)

work-list

The <work-list > element contains one <work-items> element which may contain one or more <work-item> and/or <secure-work-item> elements. The <work-list> element has the following attributes:

- `id`: specifies the unique ID for this object (required)
- `display-string`: specifies the display string for this element (required)
- `tooltip`: specifies the tooltip string. Defaults to value of `display-string` if not specified. (optional)
- `rendered`: specifies if the work list is to be rendered or not. Defaults to true. (required)

work-list Child Elements

- [icon-uri](#)
- [work-item](#)

work-lists

The <work-lists> element can contain one or more <work-list> or <secure-work-list> elements. It is a child element of <navigation-list>.

work-lists Child Elements

- [work-list](#)
- [secure-work-list](#)

Subordinate Elements**branding-uri**

The <branding-uri> element specifies the URI for the branding (logo) image. It is a child element of the <retail-portal-page> element.

home

The <home> element specifies a home work item to add to the list of homes. It is a child element of the <homes> element. The <home> element specifies a "name-value" pair that associates a role name with a work item. The <home> element has the following attributes:

- `role`: the role name (required)
- `value`: specifies the ID of the <work-item> or <secure-work-item> element that will be launched as the home page for the associated role. (required)

icon-uri

The <icon-uri> element specifies the URI that represents the icon.

resource-bundle

The <resource-bundle> element specifies the resource bundle. It is a child element of the <resource-bundles> element. The <resource-bundle> has the following attributes:

- `var` - key for the resource bundle (required)
- `resource-bundle` - name of the resource-bundle file (required)

url

The <url> element specifies the url to launch when the user clicks on this work item

value

The <value> element is a child element of the <parameter> and <custom-attribute> elements. It specifies the URL parameter value or custom attribute value.

ldap-config.xml

The `ldap-config.xml` file specifies the LDAP connection information that ORW uses to look up specific user information and administer permission grants.

The following XML elements are found in this file:

- <identity-store>—specifies the type of identity store used. This entry determines how the user's full name is looked up for display in the ORW header. Valid values are:

- **OID** – Indicates that an Oracle Internet Directory LDAP server is present.
- **v3LDAP** – Indicates that a version 3 compliant LDAP server other than OID is present.
- **XML** – Indicates that the user definitions are stored in the `system-jazn-data.xml` file or no additional user information is to be looked up in the LDAP. If **XML** is specified, then all other elements are ignored.

- `<login-dn>`—Contains the ORW application login distinguished name. ORW uses this value to log on to the LDAP. When using OID, this distinguished name must be granted proxy privileges and have the ability to look up user information. For other LDAP servers, this DN must be able to search for all the users within the LDAP's Directory Information Tree (DIT). In the LDIF scripts supplied for OID, the following entry is created with the necessary privileges:

```
orclApplicationCommonName=RetailWorkspace,cn=RetailWorkspace,cn=Products,
cn=OracleContext
```

The password for this distinguished name must be stored in the ORW wallet under the alias "ldap-user-pw". This password will be used for both OID and v3LDAP identity stores.

- `<realm-policy-dn>`—The distinguished name where the policy information is stored. This is used by the permissions management screen to read, update, and delete permission grants. The value assigned to this element must be the same as the value used by the OC4J application server. This element is only used for the OID identity stores.
- `<realm-dn>`—The distinguished name of the realm associated with the ORW application. This element is only used for the OID identity stores.
- `<realm-name>`—The realm nickname. Usually the most significant part of the `<realm-dn>` entry. For example, the **us** realm name would normally be associated with a `<realm-dn>` of `dc=us,dc=mycompany,dc=com`. This element is only used for the OID identity stores.
- `<host>`—The host name of the computer hosting the OID LDAP server. This may be a virtual IP address. This element is used for both OID and v3LDAP identity stores. This element is used for both OID and v3LDAP identity stores.
- `<port>`—A port number that the OID LDAP server is using to listen for connections. This port may be SSL or non-SSL. By default, OID listens for SSL connections on port 636 and non-SSL connections on port 389. This element is used for both OID and v3LDAP identity stores.
- `<ssl>`—A value of **true** implies the port number is for SSL connections. A value of **false** is for non-SSL connections. This element is used for both OID and v3LDAP identity stores.
- `<password-location>`—The location of the Oracle Wallet that stores the ORW application login password via the alias "ldap-user_pw". The Oracle Wallet may store additional passwords as well. This element is used for both OID and v3LDAP identity stores.
- `<user-uid-attribute>`—The LDAP attribute that uniquely identifies a user. The value of this element must match the login name of a user. This element is only used for v3LDAP identity stores.
- `<surname-attribute>`—The LDAP attribute that holds the user's surname or family name. This element is only used for v3LDAP identity stores.

- `<given-name-attribute>`—The LDAP attribute that holds the user's given name. This element is only used for v3LDAP identity stores.
- `<display-name-attribute>` -- The LDAP attribute that holds the user's display name. This element is only used for v3LDAP identity stores.

Below is an example of an `ldap-config.xml` file for use with an OID identity store:

```
<ldap-config>
  <identity-store>OID</identity-store>
  <login-dn>orclApplicationCommonName=RetailWorkspace,cn=RetailWorkspace,
cn=Products,cn=OracleContext</login-dn>
  <realm-policy-dn>cn=Policy,cn=JAZNContext,cn=products,cn=OracleContext,dc=us,
dc=oracle,dc=com</realm-policy-dn>
  <realm-dn>dc=us,dc=mycompany,dc=com</realm-dn>
  <realm-name>us</realm-name>
  <host>localhost</host>
  <port>636</port>
  <ssl>>true</ssl>
  <user-uid-attribute>uid</user-uid-attribute>
  <base-user-dn>ou=users,dc=oracle,dc=com</base-user-dn>
  <password-location>/home/oas1/product/10.1.3.2.0/OracleAS_4/j2ee/home/
RetailWorkspace/wallet</password-location>
</ldap-config>
```

Below is an example of an `ldap-config.xml` file for use with a v3LDAP that has users stored at `ou=users,dc=oracle,dc=com`. ORW will use the distinguished name `cn=ldapadmin,ou=users,dc=oracle,dc=com` to extract user information:

```
<ldap-config>
  <identity-store>v3LDAP</identity-store>
  <login-dn>cn=ldapadmin,ou=users,dc=oracle,dc=com</login-dn>
  <host>localhost</host>
  <port>636</port>
  <ssl>>true</ssl>
  <password-location>/home/oas1/product/10.1.3.2.0/OracleAS_4/j2ee/home/
RetailWorkspace/wallet</password-location>
  <user-uid-attribute>uid</user-uid-attribute>
  <base-user-dn>ou=users,dc=oracle,dc=com</base-user-dn>
  <surname-attribute>sn</surname-attribute>
  <given-name-attribute>givenname</given-name-attribute>
  <display-name-attribute>displayname</display-name-attribute>
</ldap-config>
```

Below is an example of an `ldap-config.xml` file that can be used with an identity store of the `system-jazn-data.xml` file or when ORW will not establish a connection to the LDAP:

```
<ldap-config>
  <identity-store>XML</identity-store>
</ldap-config>
```

Logging

Jakarta Commons Logging

ORW components use the Jakarta Commons Logging package for logging trace, debug, and error messages. Jakarta Commons logging provides an ultra-thin bridge between different logging libraries, enabling the ORW application to remain reasonably pluggable with respect to different logger implementations. Objects in ORW that require logging functionality maintain a handle to a Log object, which adapts logging requests to the runtime configurable logging provider.

There are two common logging implementations used with Oracle Application Server—Log4J Logging and OC4J Logging. An application server may use the OC4J Logging implementation for the container produced log entries and the ORW application may use the Log4J Logging implementation.

Additional information about Jakarta Commons Logging can be found at the following Web sites:

- <http://jakarta.apache.org/commons/logging/>
- <http://jakarta.apache.org/commons/logging/api/index.html>

Log4J Logging

Log4J loggers are one implementation available to the ORW components. This implementation is configured via a file named `log4j.xml`. A default `log4j.xml` file is deployed with ORW directing all the error messages to the console log, which results in log entries in the OPMN log file. This configuration may be overridden by adding a new `log4j.xml` to the following directory:

```
$ORACLE_HOME/j2ee/<OC4J instance>/RetailWorkspace
```

A sample `log4j.xml` configuration file is shown below. It logs the debug and higher priority messages to the file, `RetailWorkspace.log`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

  <appender name="file-appender" class = "org.apache.log4j.RollingFileAppender" >
    <param name="file" value="RetailWorkspace.log" />
    <param name="MaxFileSize" value="10MB" />
    <param name="MaxBackupIndex" value="5"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d %-5p [%t] (%F:%L) - %m%n"/>
    </layout>
  </appender>

  <root>
    <priority value="debug"/>
    <appender-ref ref="file-appender"/>
  </root>
</log4j:configuration>
```

Additional information on the Log4J Logging implementation can be found at the following Web site:

<http://logging.apache.org/log4j/1.2/index.html>

OC4J Logging

Another logging mechanism used by Oracle Application Server and OC4J is OC4J loggers. The logging levels for the application can be controlled by the Enterprise Manager administration application. Additional control of these loggers can be done via the file,

```
$ORACLE_HOME/j2ee/<OC4J instance>/config/j2ee-logging.xml.
```

The following types of loggers may be of interest to an administrator or developer:

- oracle.adf.share.*—ADF specific loggers.
- oracle.retail.*—ORW application specific loggers
- oracle.j2ee.*—Oracle application server loggers.

Note that there may be multiple loggers involved in a single over-arching area. For example, one may want to enable a **FINE** level of logging to both the oracle.j2ee.security and oracle.adf.share.security loggers.

See the *Oracle Application Server Administration Guide* for more details.

Internationalization

Internationalization is the process of creating software that can be translated more easily. Changes to the code are not specific to any particular market. ORW has been internationalized to support multiple languages.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages

The following components are not translated:

- Documentation (for example, online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Reports
- Demonstration data
- Training materials

The user interface for ORW has been translated into:

- German
- French
- Spanish

- Japanese
- Traditional Chinese
- Simplified Chinese
- Korean
- Brazilian Portuguese
- Russian
- Italian

Set the Client to the Applicable Locale

For a client machine to use a translation, the browser's language must be set. In Internet Explorer, choose the desired language from Tools > Internet Options.

Adding a New Resource Bundle to Retail Workspace

To internationalize additional custom Strings for the ORW application, it is required that a new default resource properties file be created, as well as an additional properties file for each supported locale that is needed. This new properties file is a flat text file containing name=value pairs, where the name is an abstract identifier of a resource, and the value is the actual value that is displayed at run time. For example, a new properties file called MyNewMessages.properties could contain the following name/value pair: browserTitle=New Browser Title.

This default file is locale independent, and will be used as the default resource properties file. This new file should be in a directory separate from the resource bundle properties provided by the application. An example of such directory could be:

```
com.mycompany.workspace.i18n
```

Any new additional properties files should also be placed in this newly created directory.

If specific language translation is needed, then all that needs to be done is to duplicate the MyNewMessages.properties file with a new file name that has the appropriate suffix for one of the supported languages (for example, MyNewMessages_ ja.properties for the Japanese version). You then translate the values (leaving the names of the resources the same).

To register the new resource bundle with the application, the bundle must be defined in the retail-workspace-page-config.xml configuration file. A 'new' <resource-bundle> element needs to be defined under the <resource-bundles> element (refer to the Element Definition Documentation for more information). The example below defines the new MyNewMessages bundle referenced earlier.

```
<resource-bundle var="unique_identifier"  
  
resource-bundle="com.mycompany.workspace.i18n.MyNewMessages" />
```

Once defined, the new custom message bundle is referenced by EL expressions. The 'var' parameter specifies the EL 'prefix' for the bundle. It must be unique in this configuration file and cannot duplicate another 'var' specified in any <load-bundle> tags used elsewhere in the application

It is important to note that only the default bundle needs to be defined in the configuration file. The application automatically uses the correct bundle for other locales based on the user's browser locale setting. If the application cannot find the

bundle based on the user's locale, then the default bundle is used, in this case `MyNewMessages.properties`.

Once the bundles are loaded, the message strings can be accessed using EL expressions.

For example: `#{unique_identifier.browserTitle}`

Deployment of New Resource Bundle(s)

The newly created resource bundle(s) must be copied into the `RetailWorkspace/lang_packs` directory where the ORW application is installed (`($ORACLE_HOME/j2ee/<instance name>/RetailWorkspace/lang_packs/)`).

Choose one of the following two options for placing the messages file(s) in `RetailWorkspace/lang_packs`:

1. Below the `lang_packs` directory, create a directory structure that matches the package name of the resource bundle, i.e. `com/mycompany/workspace/i18n`.
Copy `MyNewMessages.properties` and `MyNewMessages_ja.properties` to the newly created `i18n` directory.
2. Package the resource bundle files in a jar file, maintaining the `com/mycompany/workspace/i18n` directory structure within the jar file.
Copy this jar file to the `lang_packs` directory.

Maintenance

Exporting and Importing Portlet Customizations

Oracle Retail Workspace is a WebCenter based application with support for customization and personalization of the portlets. Portlet customizations are default preferences that are visible to all users. Portlet personalizations are preferences that are visible only to the user who specified them. With the right permissions, users of ORW can customize and personalize portlets on dashboards and other ORW pages.

Portlet customizations and personalizations are stored in the Metadata Services (MDS) repository. With this capability comes the need for migrating portlet preferences from one environment to another. For example, one might want to migrate customizations from a development environment to a production environment, from one production environment to another, or from one ORW version to another.

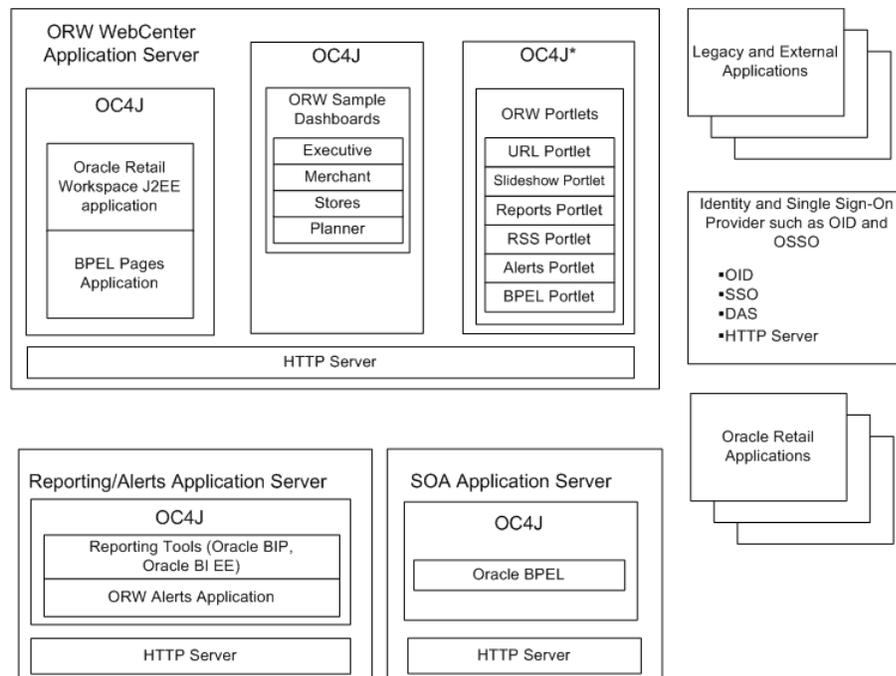
ORW provides tools for exporting and importing portlet customizations. Export and import of portlet personalizations are not supported. For more information, see the section *Exporting and Importing Customizations* in the chapter *Managing Workspace* of the *Oracle Retail Workspace Administration Guide*.

Technical Architecture

This chapter describes the overall software architecture for ORW and provides an overview of the enterprise infrastructure software used along with it. A high-level discussion of the general structure of the systems is included. From this content, integrators can learn about the parts of the application and the interaction between the parts.

Oracle Retail Workspace Technical Architecture Diagram and Descriptions

Figure 3–1 Oracle Retail Workspace Technical Architecture



* includes additional Portlet libraries

Oracle Retail Workspace Architecture

ORW is a JavaServer Faces (JSF) Java Enterprise Edition (Java EE) application that provides a single point of entry to participating Oracle Retail applications. In addition, ORW integrates with and provides navigation to various types of content including:

- Reports created with Oracle Business Intelligence Enterprise Edition (BI EE) and Oracle Business Intelligence Publisher (BIP)
- Oracle BI Delivers Alerts
- BPM Worklists and BPEL Workflows

ORW is comprised of the following components:

- The Oracle Retail Workspace (ORW) application
- A portlet application bundle that includes seven re-usable portlets
- Four sample dashboard applications
- A ORW Alerts application deployed to the BI EE server.
- A BPEL Pages application

Each of these components is packaged and deployed as a separate Java EE enterprise archive (EAR).

ORW components are deployed to Oracle Containers for Java EE (OC4J). OC4J is the core Java EE runtime component of Oracle Application Server. ORW requires Oracle Application Server 10g Release 3 (10.1.3.4).

The ORW user interface, sample dashboards and portlets were all developed using Oracle JDeveloper (10.1.3.4). The user interface and sample dashboards were developed using Oracle ADF Faces JSF components. ADF Faces is a set of user interface components based on the JavaServer Faces JSR (JSR-127). In addition, ADF Faces also provides a number of JSF application framework features. ORW takes advantage of a number of these features, including:

- Partial page rendering
- Internationalization and accessibility support
- Look and feel "skinning" support

ORW also requires the Oracle WebCenter Framework component of Oracle WebCenter Suite 10g Release 3 (10.1.3.4). WebCenter Suite is a suite of technologies that brings together standards-based development using JavaServer Faces with the flexibility and power of portals, to provide the ability to build context-rich applications. The WebCenter Framework makes it possible to consume JSR-168 standards based portlets and Oracle PDK portlets within a JSF application.

ORW can be configured to use an external LDAP server to provide its user repository and such a configuration must be used within a Single Sign-On system. ORW may be configured to use a number of implementations for this, but only Oracle Internet Directory (OID) and Oracle Single Sign-On (OSSO) are officially supported. OID is an LDAP directory that takes advantage of the scalability, high availability, and security features of Oracle Database. OID and Oracle SSO are provided by Oracle Identity Management 10gR2 (10.1.2), which is included in Oracle Application Server Infrastructure 10g Release 2 (10.1.2.0.2). OID serves as the central user repository for Oracle Identity Management. It simplifies user administration in ORW and provides a standards-based application directory for the heterogeneous enterprise. Additionally, Oracle Directory Synchronization allows Oracle Identity Management to seamlessly

integrate with other directories and enterprise user repositories, enabling identity information for users to be accessed wherever it resides.

ORW includes support for a Tools work list that provides ORW users direct links to Oracle Delegated Administration Service units. Delegated Administration Service (DAS) is a web based GUI that is used to create and manage users and groups in OID. The DAS application is only provided with OID.

The ORW user interface is a highly configurable application that allows customers to develop their own content using JDeveloper, ADF Faces and the WebCenter Framework. The ORW navigation panel and content may be configured using an XML configuration file. The default ORW navigation panel includes a set of six work lists. Customers are free to delete or change the configuration of existing work lists, add new work lists, and assign role-based "home pages". Retailers may control access to work list content by securing items in the work list or the work list itself and granting access permissions to specific roles (OID groups).

The ORW portlet application includes a set of seven re-usable JSR-168 portlets:

- Alerts Portlet - a portlet that may be used to display Oracle BI EE Ibot Alerts.
- URL Portlet - a simple portlet that may be used to display the contents of a web page.
- Reports Portlet - a specialized version of the URL Portlet that may be used to display reports that have been developed using Oracle BI EE or BIP, or other reporting tools that expose reports through URLs.
- Slideshow Portlet - a portlet that may be used to display the contents of several Web pages in a slideshow fashion.
- RSS Portlet - a portlet that may be used to display the contents of an RSS feed that conforms to RSS 2.0.
- BPM Worklist portlet - a portlet that may be used to view and manage a user's tasks from the BPEL workflows.
- BPEL Workflow portlet - a portlet that may be used to view and invoke a user's role specific BPEL workflows.

The example dashboard applications were developed using JDeveloper, ADF Faces and the WebCenter Framework, and make use of the Reports Portlet and the RSS Portlet. These dashboards provide examples of how custom content may be developed for ORW.

Retail Workspace User Interface Framework

The ORW user interface includes a highly configurable framework to allow for customization of content. Retailers may develop their own dashboard content using JDeveloper, ADF Faces, and the WebCenter Framework components. Retailers are free to use any of the seven portlets included with ORW, or they may use any of the portlets packaged with the Oracle WebCenter Framework. Retailers may also develop their own JSR-168 portlets. They then may integrate the content into ORW by changing the ORW XML configuration file, retail-workspace-page-config.xml.

Retailers may also change the ORW configuration to add navigation links to other applications and web pages of their choice.

ORW takes advantage of ADF Faces support for "look and feel" skinning. ADF Faces "skins" are style sheets based on CSS 3.0 syntax that are specified in one place for an entire ADF Faces application. ADF Faces components are designed to use the styles as defined in the skin. The ORW user interface and dashboards are deployed with a

custom ADF Faces skin. In addition to its default skin, ORW provides an example skin that may be used as the basis for creating a custom skin. Retailers are free to customize the application's look and feel by developing their own skin based on the example skin.

Single Sign-on Overview and Topology

What is Single Sign-On?

Single Sign-On (SSO) is a term for the ability to sign onto multiple web applications via a single user ID/Password. There are many implementations of SSO - Oracle currently provides three different implementations: Oracle Single Sign-On (OSSO), JavaSSO, and Oracle Access Manager (provides more comprehensive user access capabilities).

Most, if not all, SSO technologies use a session cookie to hold encrypted data passed to each application. The SSO infrastructure has the responsibility to validate these cookies and, possibly, update this information. The user is directed to log on only if the cookie is not present or has become invalid. These session cookies are restricted to a single browser session and are never written to a file.

Another facet of SSO is how these technologies redirect a user's Web browser to various servlets. The SSO implementation determines when and where these redirects occur and what the final screen shown to the user is.

Most SSO implementations are performed in an application's infrastructure and not in the application logic itself. Applications that leverage infrastructure managed authentication (such as deploying specifying "Basic" or "Form" authentication) typically have little or no code changes when adapted to work in an SSO environment.

What Do I Need for Oracle Single Sign-On?

The nexus of an Oracle Single Sign-On system is the Oracle Identity Management Infrastructure installation. This consists of the following components:

- An Oracle Internet Directory (OID) LDAP server, used to store user, role, security, and other information. OID uses an Oracle database as the back-end storage of this information.
- An Oracle Single Sign-On servlet, used to authenticate the user and create the OSSO session cookie. This servlet is deployed within the infrastructure Oracle Application Server (OAS).
- The Delegated Administration Services (DAS) application, used to administer users and group information. This information may also be loaded or modified via standard LDAP Data Interchange Format (LDIF) scripts.
- Additional administrative scripts for configuring the OSSO system and registering HTTP servers.

Additional OAS servers will be needed to deploy the business applications leveraging the OSSO technology.

Can Oracle Single Sign-On Work with Other SSO Implementations?

Yes, OSSO has the ability to interoperate with many other SSO implementations, but some restrictions exist.

Can Oracle Retail Workspace Work with Other SSO Implementations?

Yes, ORW can be used with other SSO implementations, assuming they are compatible with Oracle Application Server v10.1.3.4. However, ORW only officially supports OSSO. Any problems encountered enabling ORW to work within a non-supported SSO implementation must be directed to the SSO vendor. This guide provides information on the Oracle Single Sign-On as used in context of ORW.

Oracle Single Sign-on Terms and Definitions

Authentication

Authentication is the process of establishing a user's identity. There are many types of authentication. The most common authentication process involves a user ID and password.

Dynamically Protected URLs

A "Dynamically Protected URL" is a URL whose implementing application is aware of the OSSO environment. The application may allow a user limited access when the user has not been authenticated. Applications that implement dynamic OSSO protection typically display a "Login" link to provide user authentication and gain greater access to the application's resources.

Identity Management Infrastructure

The Identity Management Infrastructure is the collection of product and services which provide Oracle Single Sign-on functionality. This includes the Oracle Internet Directory, an Oracle HTTP server, and the Oracle Single Sign-On services. The Oracle Application Server deployed with these components is typically referred as the "Infrastructure" instance.

MOD_OSSO

mod_osso is an Apache Web Server module an Oracle HTTP Server uses to function as a partner application within an Oracle Single Sign-On environment. The Oracle HTTP Server is based on the Apache HTTP Server.

Oracle Internet Directory

Oracle Internet Directory (OID) is an LDAP-compliant directory service. It contains user ids, passwords, group membership, privileges, and other attributes for users who are authenticated using Oracle Single Sign-On.

Partner Application

A partner application is an application that delegates authentication to the Oracle Identity Management Infrastructure. One such partner application is the Oracle HTTP Server (OHS) supplied with the Oracle Application Server. OHS uses the MOD_OSSO module to configure this functionality.

All partner applications must be registered with the Oracle Single Sign-On server. An output product of this registration is a configuration file the partner application uses to verify a user has been previously authenticated.

External Application

An external application is an application not integrated with the Oracle Identity Management Infrastructure. These applications have their own authentication and authorization facilities. They may even have their own user identity and security policy storage.

Oracle Identity Management provides a means to store and submit user credentials for external applications. See the section, "External Application Integration", later in this chapter for more details.

Realm

A Realm is a collection of users and groups (roles) managed by a single password policy. This policy controls what may be used for authentication (for example, passwords, X.509 certificates, biometric devices). A Realm also contains an authorization policy used for controlling access to applications or resources used by one or more applications.

A single OID can contain multiple Realms. This feature can consolidate security for retailers with multiple banners or to consolidate security for multiple development and test environments.

Statically Protected URLs

A URL is considered to be "Statically Protected" when an Oracle HTTP server is configured to limit access to this URL to only SSO authenticated users. Any attempt to access a "Statically Protected URL" results in the display of a login page or an error page to the user.

Servlets, static HTML pages, and JSP pages may be statically protected.

What Single Sign-On is not

In general, Single Sign-On is NOT a user ID/password mapping technology. However, Oracle Single Sign-On does provide a facility for this. Additionally, some applications can store and retrieve user IDs and passwords for non-SSO applications within an OID LDAP server. An example of this is the Oracle Forms Web Application framework, which maps OSSO user IDs to a database logins on a per-application basis.

How Oracle Single Sign-On Works

Oracle Single Sign-On involves multiple components. These are:

- The Oracle Single Sign-On (OSSO) servlet, which is responsible for the back-end authentication of the user.
- The Oracle Internet Directory LDAP server, which stores user IDs, passwords, and group (role) membership.
- The Oracle HTTP Server associated with the web application, which verifies and controls browser redirection to the OSSO servlet.
- If the web application implements dynamic protection, then the web application itself is involved with the OSSO system.

Statically Protected URLs

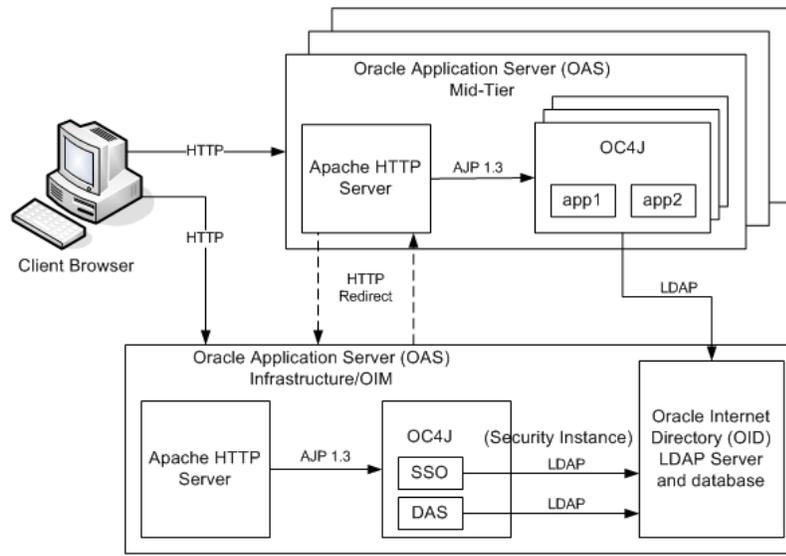
When an unauthenticated user accesses a statically protected URL, the following occurs:

1. The Oracle HTTP server recognizes the user has not been authenticated and redirects the browser to the Oracle Single Sign-On servlet.
2. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
3. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple Realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
4. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.
5. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
6. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

Dynamically Protected URLs

When an unauthenticated user accesses a dynamically protected URL, the following occurs:

1. The Oracle HTTP server recognizes the user has not been authenticated, but allows the user to access the URL.
2. The application determines the user must be authenticated and sends the Oracle HTTP server a specific status to begin the authentication process.
3. The Oracle HTTP Server redirects the user's browser session to the OSSO Servlet.
4. The OSSO servlet determines the user must authenticate him/herself, and so displays the OSSO login page.
5. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
6. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.
7. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
8. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

Figure 3–2 Single Sign-on Topology

OSSO Installation Overview

Installing Oracle Single Sign-On consists of installing the following components:

1. Installing the Oracle Internet Directory (OID) LDAP server and the Infrastructure Oracle Application Server (OAS). These are typically performed using a single session of the Oracle Universal Installer for the Infrastructure OAS 10g (10.1.2.2) release. OID requires an Oracle relational database which, if not available, can also be installed in the same installer session.

The Infrastructure OAS includes the Delegated Administration Services (DAS) application as well as the OSSO servlet. The DAS application can be used for user and realm management within OID.

2. Installing additional OAS 10.1.2 mid-tier instances for the Oracle Retail applications, such as RMS, that are based on Oracle Forms technologies. These instances must be registered with the Infrastructure OAS installed in step 1).
3. Installing additional application servers to deploy other Oracle Retail applications and performing application specific initialization and deployment activities.

Infrastructure Installation and Configuration

The Infrastructure installation for OSSO is dependent on the environment and requirements for its use. Deploying an Infrastructure OAS to be used in a test environment does not have the same availability requirements as for a production environment. Similarly, the Oracle Internet Directory (OID) LDAP server can be deployed in a variety of different configurations. See the *Oracle Application Server Installation Guide* and the *Oracle Internet Directory Installation Guide* for more details.

OID User Data

Oracle Internet Directory is an LDAP v3 compliant directory server. It provides standards-based user definitions out of the box.

The current version of Oracle Single Sign-On only supports OID as its user storage facility. Customers with existing corporate LDAP implementations may need to synchronize user information between their existing LDAP directory servers and OID.

OID supports standard LDIF file formats and provides a JNDI compliant set of Java classes as well. Moreover, OID provides additional synchronization and replication facilities to integrate with other corporate LDAP implementations.

Each user ID stored in OID has a specific record containing user specific information. For role-based access, groups of users can be defined and managed within OID. Applications can thus grant access based on group (role) membership saving administration time and providing a more secure implementation.

OID with Multiple Realms

OID and OSSO can be configured to support multiple user Realms. Each realm is independent from each other and contains its own set of user IDs. As such, creating a new realm is an alternative to installing multiple OID and Infrastructure instances. Hence, a single Infrastructure OAS can be used to support many development and test environments by defining one realm for each environment.

Realms may also be used to support multiple groups of external users, such as those from partner companies. For more information on Realms, see the *Oracle Internet Directory Administrators Guide*.

User Management

User Management consists of displaying, creating, updating or removing user information. There are two basic methods of performing user management in Oracle Identity management: LDIF scripts and the Delegated Administration Services (DAS) application.

OID DAS

The DAS application is a web based application designed for both administrators and users. A user may update his or her own password, change his or her telephone number of record, or modify other user information. Users may search for other users based on partial strings of the user's name or ID. An administrator may create new users, unlock passwords, or delete users.

The DAS application is fully customizable. Administrators may define what user attributes are required, optional or even prompted for when a new user is created.

Furthermore, the DAS application is secure. Administrators may also define what user attributes are displayed to other users. Administration is based on permission grants, so different users may have different capabilities for user management based on their roles within their organization.

LDIF Scripts

Script based user management can be used to synchronize data between multiple LDAP servers. The standard format for these scripts is the LDAP Data Interchange Format (LDIF). OID supports LDIF scripts for importing and exporting user information. LDIF scripts may also be used for bulk user load operations.

User Data Synchronization

The user store for Oracle Single Sign-On resides within the Oracle Internet Directory (OID) LDAP server. Oracle Retail applications may require additional information attached to a user name for application-specific purposes and may be stored in an application-specific database. Currently, there are no Oracle Retail tools for synchronizing changes in OID stored information with application-specific user stores. Implementors should plan appropriate time and resources for this process. Oracle

Retail strongly suggests that you configure any Oracle Retail application using an LDAP for its user store to point to the same OID server used with Oracle Single Sign-On.

External Application Integration

Oracle Identity Management's Single Sign-On implementation includes a facility for integration with external applications. This facility allows an administrator to define the parameters an application uses to authenticate a user and/or launch the application. Once the application is defined, a user's username/password can be securely stored and used when the user desires to launch the application. Oracle Retail Workspace provides a link to the appropriate URL for managing external application definitions.

Note: A user must be a member of the iASAdmins group in order to manage external applications.

The mechanism used for authenticating a user to an external application typically involves redirecting the browser to a specific URL or submitting a form to the external application. Parameters on the form or redirected URL contain the user's username/password. This process is stateless: it only performs the redirect once and does not remember the state of previous attempts to access the application.

However, some applications require more complex interactions with the user's browser. These applications store state information in the browser or in the application server and use this information as part of the login process. The first time a browser accesses the application, a login screen appears. But on the second and subsequent accesses the login screen will not appear. The OIM external application facility cannot by itself allow a user to transparently login to the external application. One solution to this problem is to use JavaScript to provide the transparent login. Oracle Retail Workspace has provided sample HTML and JavaScript an installation may leverage for this purpose.

External Applications

Any application that does not participate in the Oracle Single Sign-On authentication process is known as an External Application. These applications typically implement their own authentication process. As such, the user is normally prompted for an application specific user name and password.

This chapter details how to provide a Single Sign-On experience when using these applications and Oracle Retail Workspace.

Oracle Single Sign-On and External Applications

Oracle Single Sign-On (OSSO) includes a facility for transparently logging into External Applications. This facility requires the user to enter his/her credentials the first time the application is accessed and then stores/retrieves this information for use later. This facility is an integral part of the Oracle Portal Development Kit (PDK), but may be used outside of the PDK, since many of the OSSO External Application API's are exposed as URLs.

The following steps are followed when using the OSSO External Application facilities:

1. An administrator creates an External Application Definition in OSSO. This definition includes the URL to submit the user's credentials and any other parameters needed to invoke the external application.
2. A user defines his/her credentials (typically a username and password) to use when accessing the External Application. This information is encrypted and stored in the OSSO database.
3. A user running a Web application invokes a link to the OSSO External Application "process login" URL. A parameter on this URL specifies which External Application the user is accessing.
4. The OSSO External Application facility reads the External Application definition and the user's credentials for that application. It then creates a URL containing the credentials. Depending on the External Application definition, the user's browser is then either redirected to this URL or submits a form to this URL.
5. The External Application processes the request made in step 4, and the user is logged into the application.

The OSSO External Application facility does not maintain any state for the user's browser session. OSSO does not know whether the user has already logged into the External Application or not. Consequently, the OSSO External Application facility is only concerned about a single operation - submitting the user's credentials to a specific URL.

However, some applications require additional processing. For example, an application may require a browser to request a specific URL before accessing the URL used to authenticate the user. Moreover, this second URL may only be legally accessed if and only if the user has not previously logged into the External Application. A category of these applications uses the "j_security_check" authentication mechanism. Sample JavaScripts are provided that may be used to provide a Single Sign-On experience with these applications.

OSSO Prerequisites

When OSSO is configured to use an Oracle 10.2.0 or later database, the "init.ora" configuration file must contain a specific "event" entry in order for the External Applications facility to work. The specific configuration information is documented in My Oracle Support Document #344602.1.

Supported Types of Authentication

The OSSO External Application facility supports three mechanisms for submitting a user's credentials.

1. BASIC Authentication. The user's username and credentials are supplied to the application as part of the URL. For example,

```
http://username:password@host.company.com/appname
```

Note: URLs containing the "username:password" construct are not supported in recent versions of Microsoft's Internet Explorer unless specific Windows Registry entries are modified. See Microsoft's web site for more details (Help and Support article #834489). Another alternative is to access the External Application via a JavaScript, such as that used by basicAppRemote.html, described later.

2. GET Authentication. The user's user name and password are passed as parameters in the URL's query string and an HTTP GET operation is performed. The names of these parameters are application specific. The query string may be seen in the web browser's address bar when the operation has completed.
3. POST Authentication. The user's user name and password are passed as parameters within an HTTP POST (form submit) operation. The credentials information is not displayed to the user.

Security Considerations Launching External Applications

The main security consideration for launching an external application is protecting the user names and passwords from disclosure. OSSO stores this information in an encrypted form in the OSSO database. However, unless the URLs accessed use the secure HTTPS protocol, this information is not encrypted when the browser submits the user's credentials to the external application. Unless HTTPS is used with BOTH the OSSO URLs and the External Application's URLs, the user's credentials are transmitted in clear text or in an easily decoded format. Furthermore, if "GET" authentication is used, this information may also be displayed in the browser's address bar.

Defining an External Application in OSSO

Required Privileges

An administrator must be a member of the iASAdmins group in the OID LDAP in order to manage OSSO External Application definitions. Members of this group have significant additional privileges as well.

The user ID, 'orcladmin' is typically made a member of the iASAdmins group when OID is installed. The DN of the iASAdmins group is

```
cn=iASAdmins,cn=Groups,cn=OracleContext
```

Adding members to this group may be performed using the ldapadd command, the oidadmin tool or other LDAP browser.

Launching OSSO External Application Management

Retail Workspace can be configured to provide a link to the OSSO URL used to manage external applications. The OSSO URL for managing external applications has the following form:

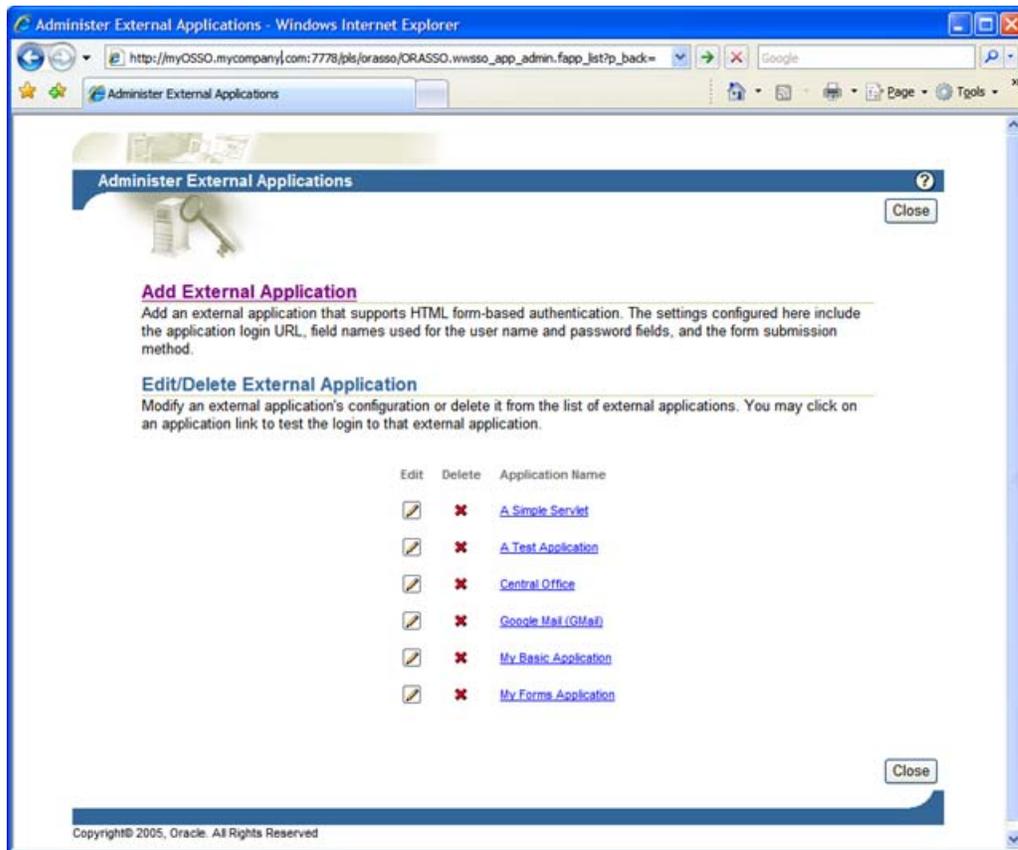
```
http[s]://[OSSO Host]:[OSSO Port]/pls/orasso/ORASSO.wvssso_app_admin.fapp_list
```

where:

- http[s] is either "http" or "https"
- [OSSO Host] is the hostname of the OSSO Server
- [OSSO Port] is the OSSO server port number

A link to this URL is available in the sample retail-workspace-page-config.xml file available upon Oracle Retail Workspace product installation. The link is found under the "Admin Tools" folder in the "Tools" worklist.

Figure 4-1 The main OSSO External Application Administration page



To add a new External Application, click on the "Add External Application" link. This will bring up the page found in Figure 4-2.

Figure 4–2 Creating a new External Application definition in OSSO

External Application Login
 Enter the application name, the login URL, and the user name and password HTML field names used by the application's login form. The login URL is typically the submit action of the application's login form. It will be used in conjunction with the user name and password field names to perform a single sign-on login into this application. The login URL as well as the user name and password field names should be determined by inspecting the source of the application's standard login form. User name/id, password, additional field etc. values are not required for Basic authentication and Login URL should be a url which requires authentication.

Application Name:
 Login URL:
 User Name/ID Field Name:
 Password Field Name:

Authentication Method
 Select the authentication method used by this application. The POST method submits the credentials with the body of the form. The GET method submits the login credentials as part of the login URL.

Type of Authentication Used:

Additional Fields
 Type the names and values of any additional fields that are submitted with the login form of the external application.

Field Name	Field Value	Display to User
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>

A detailed description of these fields can be found in the Configuring and Administering External Applications chapter of the *Oracle Application Server Single Sign-On Administrator's Guide*. A brief description is below:

- **Application Name:** Name that appears on the main External Application Management page.
- **Login URL:** The URL (host name, port, URI) used to submit the user's credentials.
- **User Name/ID Field Name:** The name of the parameter which identifies the user name or user ID. The value of this parameter is the user name used to log into the external application. This is valid only for the "GET" and "POST" authentication types.
- **Password Field Name:** The name of the parameter which identifies the password used to log into the external application. This is valid only for the "GET" and "POST" authentication types.
- **Type of Authentication Used:** The External Application's login mechanism. This is a drop-down menu.
- **Additional Fields:** This is a table of additional parameters associated with the Login URL. These parameters, along with the UserName/ID Field and the Password Field, will be put into the Login URL's query string (BASIC or GET authentication) or submitted as form input parameters (POST authentication). The third column of this table, "Display to User", contains a check box. If checked, then the field value may be altered by the user when the user enters his/her credentials for the application.

Note: A user's credentials can be defined the first time the External Application is accessed. However, this capability may be overridden by other factors.

Determining the Authentication Type

To determine the authentication type, the first step is to access the main application page as an unauthenticated user. External Applications using BASIC authentication will cause the Web browser to display a pop-up login dialogue box. An example of Microsoft's Internet Explorer version 7 dialog box is shown in [Figure 4-3](#) below.

The presence of a pop-up log-in dialogue does not insure BASIC authentication is used. However, if one does appear and BASIC authentication is not used, then the External Application may not be able to be supported as an OSSO External Application.

An application may indicate GET or POST authentication by presenting the ability to login as a form or link on the main application page. To determine if POST authentication is used, use the browser's menu option to view the page source. Look for the <FORM> tag containing the button used to log in. If this tag contains an attribute labeled "method" with a value of "GET", then GET authentication is most likely used. Otherwise, POST authentication is probably used. Note that other factors may also influence which authentication type is actually used, such as the execution of JavaScript when the login information is submitted.

Note that the <FORM> element may contain an "action" attribute. This attribute should be used as the Login URL.

Figure 4-3 Microsoft Internet Explorer version 7 BASIC Authentication Pop-up dialogue



Determining Field Names and Values

If BASIC authentication is used, then the values of "User Name/ID Field Name" and "Password Field Name" should be left blank. To determine the names and values of the "Additional Fields" table, look at the URL used to access the application and examine the query string parameters, if any. The query string is that part of the URL after the question mark (?). Each query string parameter is in the form [field name]=[field value] and is separated by an ampersand (&).

If GET authentication is used, then the field names and/or values may be determined by the URL in the address bar seen immediately after logging in. To determine the "User Name/ID Field Name", look for a query string parameter that specifies the user name you just used to log into the application. To determine the "Password Field Name" look for a query string parameter specifying the password value used. The remaining parameters should go into the "Additional Fields" table.

For example, suppose GET authentication is used and a user can log in with the user name of "user123" and a password of "xxx1234AS" and the URL used for logging into the app is in the following form:

```
http://ahost.mycompany.com/myapp?userid=user123&pswd=xxx1234AS&p1=54&lswiz=wonder
```

Then the "User Name/ID Field Name" should be set to "userid" and the "Password Field Name" should be set to "pswd". Also, two fields should be put into the "Additional Fields" table, with the field names of "p1" and "lswiz" and with values of "54" and "wonder", respectively.

The URL used to log into the application using GET authentication may not be available on the browser address bar. If this is the case, then look for input fields on the form and follow the same general procedure listed below for POST authentication.

The parameters submitted with an application using POST authentication never appear in the browser's address bar. One way to determine the field names and values is to examine the source of the page via the browser's capability to view the HTML source of the page containing the login form.

Examine this page and find the <FORM> tag containing the <INPUT> elements for the user name and the password. These <INPUT> elements map to the "User Name/ID Field Name" and the "Password Field Name" in the External Application definition in OSSO. Each <INPUT> element should contain a "name" attribute. The "name" attribute for the <INPUT> element for the user name is the value of the "User Name/ID Field Name" in the External Application definition. The "name" attribute for the <INPUT> element for the password is the "Password Field Name".

All other <INPUT> elements in this <FORM> element are the "Additional Fields" table, where the "name" attribute maps to the "Field Name" column and any "value" values map to the "Field Value" column.

Example:

```
<form action = "https://myhost.mycompany.com/myapp/login" onsubmit= "preLogin()"
method= "POST" >
<table border="0">
  <tr>
<td>User Name</td><td><input type="text" name="userid" /></td>
</tr>
<tr>
<td>Password</td><td><input type= "password" name= "pwd" /></td>
</tr>
</table>
<input type= "hidden" name = "usertype" value= "normal" />
<input type= "hidden" name = "webaccess" value= "true" />
<input type= "hidden" name = "verbosity" value= "verbose" />
</form>
```

In the example above:

- Login URL is "https://myhost.mycompany.com/myapp/login"
- Authentication Type is "POST"

- User Name/ID Field Name is "userid"
- Password Field Name is "pwd"
- There are three additional fields named "usertype", "webaccess", and "verbosity" with values of "normal", "true" and "verbose", respectively.

Alternate Methods Determining OSSO External Application information

Besides looking at the source HTML associated with an external application, an administrator may choose to use a browser plug-in or a packet sniffing application to determine this information. These tools are sold by a variety of corporations and are also available free from some Open Source projects. Examples of Open Source providers include the "Firebug" plug-in for the Mozilla Firefox browser and the "WireShark" packet sniffer tool.

These tools have their advantages and disadvantages. Packet sniffers look at information as it is presented to the network. When the HTTPS protocol is used, packet sniffers cannot determine the parameters used, since this information is encrypted. Browser plug-ins may also have problems remembering complex interactions between the browser and the application.

Testing an External Application Definition

To determine if the information contained in an External Application definition is correct, click on the application name link found in the main OSSO External Application Administration page (Figure 1). Clicking on a link will launch the application in a new window using the current definition.

Note: A user's credentials can be defined the first time the External Application is accessed.

Some applications use a special "login" URL and also require the user to access the main application page first. The "login" URL only becomes valid when a non-authenticated user accesses the main application page. These applications are referred to in this document as, "j_security_check" applications. A "j_security_check" application must use additional resources besides the OSSO External Application facility in order to accommodate transparent user-id and password mapping. Details on this is found later in this document.

An example of a "j_security_check" application is the Central Office application.

Obtaining the OSSO External Application ID

Each External Application definition has a unique identifier. This identifier is presented in many of the URLs within OSSO used to launch the application or manage the application's definition within OSSO. As such it is required when launching the application from Retail Workspace, or to allow a user to create/modify credentials for this application.

The OSSO External Application ID may be obtained from the p_app_id parameter found in many of the OSSO URLs. Figure 4 shows an OSSO External Application with an ID of B9393469CEC51BDBBD9A8FA37DCDACB2. The specific example in Figure 4 was the result of clicking on the "Edit" link in the main OSSO External Application Administration page.

Figure 4–4 The p_app_id parameter containing the OSSO External Application ID

Application Login

ation name, the login URL, and the user name and password HTML field names used by the application's
 configuration of the application's login form. It will be used in combination with the user name and password of

Managing User Credentials

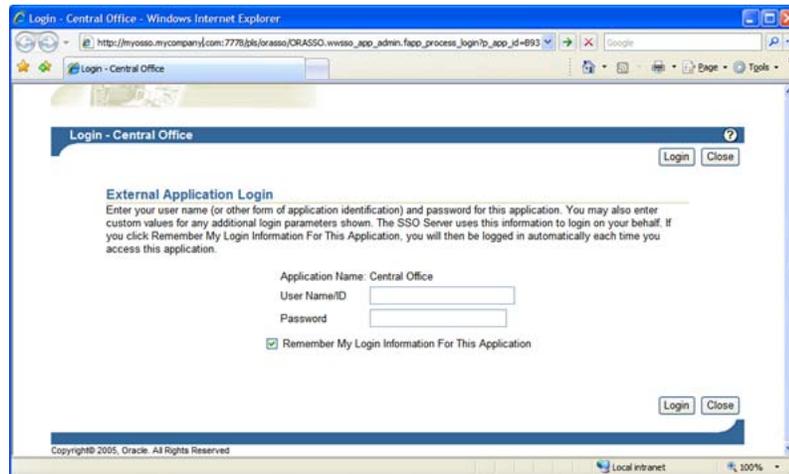
User credentials are managed on a per application basis. A user is presented with the "External Application Login" page (Figure 5) when attempting to launch an application before defining his/her credentials to use for that application. The URI and query string segments of the URL must be `/pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser?p_app_id=[OSSO ID]`.

Where [OSSO ID] is the OSSO External Application ID. The host/port/scheme portions of the URL are the same as used with DAS.

You may leverage the DAS URL managed bean to find the OSSO host/port/scheme segments when configuring the `retail-workspace-page-config.xml` file. The example below shows an entry used to manage user credentials for Central Office:

```
<secure-work-item id="userCredCentralOffice"
  display-string="Central Office (CO)" rendered="true"
  launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>B9393469CEC51BDBBD9A8FA37DCDACB2</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Figure 4–5 The External Application Login page used to store credentials for a user



Configuring Oracle Retail Workspace to Launch External Applications

This section details the entries in the Retail Workspace configuration file, `retail-workspace-page-config.xml`, needed for launching and managing external applications.

Note: This section does not cover using the sample scripts `basicAppRemote.html`, `jSecAppRemote.html`, and `jSecAppORW.html`. See the section, "Retail Workspace Sample External Application JavaScripts", for details on using these scripts.

All of the examples use the `<secure-work-item>` element. They will require new `WorkElement` permission grants before any user can access these entries in the Retail Workspace application. Permission grants can be created in the Permissions Management page of Retail Workspace.

Note: Permission Grants are cached by the Oracle Application Server container. This cache may take some time before it is refreshed with new information, depending on the configuration of the `jazn.xml` file. The default cache refresh period is one hour.

External Applications require three entries in the `retail-workspace-page-config.xml` file:

- **An entry to create/update/delete External Applications in OSSO.** The OSSO host/port/scheme information may be retrieved from the DAS URL managed bean found within Retail Workspace.

An example of this entry in the `retail-workspace-page-config.xml` file is:

```
<secure-work-item id="adminExternalApps"
    display-string="{confMgs.manageExternalApps}" rendered="true"
    launchable="true" show-in-content-area="false">
    <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.fapp_list</url>
</secure-work-item>
```

In the sample retail-workspace-page-config.xml file, this entry already exists in the "Admin Tools" folder found in the "Tools" worklist.

- **An entry to launch the specific External Application.** This entry will reference the OSSO External Application ID to actually launch the URL. This entry may reference the DAS URL bean to find the OSSO host/port/scheme information.

An example of this entry in the retail-workspace-page-config.xml file is:

```
<secure-work-item id="launchMyExternalApp"
display-string="Launch My External Application" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.fapp_process_
login</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>B9393469CEC51BDBBD9A8FA37DCDACB2</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Note: This entry will not launch applications using the "j_security_check" or similar authentication. Details on launching these types of applications are found later in this chapter.

- **An entry to allow a user to manage the credentials used when launching the External Application.** This entry is needed in case the user enters the wrong credentials or the credentials change because of some external event. This entry will reference the OSSO External Application ID. This entry may reference the DAS URL bean to find the OSSO host/port/scheme information.

An example of this entry in the retail-workspace-page-config.xml file is:

```
<secure-work-item id="userCredForMyExternalApp"
display-string="Manage Credentials for My External Application" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>B9393469CEC51BDBBD9A8FA37DCDACB2</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Retail Workspace Sample External Application JavaScripts

The OSSO External Application facility cannot launch all applications by itself. Furthermore, recent versions of Microsoft Internet Explorer do not support the URL supplied by OSSO for applications using BASIC authentication. As such Oracle Retail Workspace includes a set of sample HTML and JavaScript files one can leverage to achieve a Single Sign-On capability for most external applications.

Note: While these HTML and scripts can be used for many applications, some applications may require additional coding due to application, browser, network, and security issues specific to the application and its deployment. These scripts may not be appropriate for all applications or may require changes for a specific application.

The following HTML and JavaScript files are supplied with the Retail Workspace application. These scripts are found in the directory, [ORW Install Home] /workspace/workspaceapp/ExternalAppLaunch.

where [ORW Install Home] is the directory where Oracle Retail Workspace is installed.

Note: All of the sample entries below use the <secure-work-item> element in retail-workspace-page-config.xml. In order to see these entries, appropriate WorkElement permission grants will need to be created for the roles allowed to see these links in the Navigation panel. Permission grants are created via the Permissions Management administration page.

Note: All of the sample entries below specify fixed text for the work item "display-string" attribute. As such, these strings are not internationalized. To internationalize these strings, one needs to add a key=value entry in a resource bundle and reference this entry as an EL expression.

Installation

The HTML, JavaScript, and configuration files can be installed on most Apache HTTP Servers by copying the files from the files from [ORW Install Home] /workspace/workspaceapp/ExternalAppLaunch to a sub-directory of an HTTP Server's "htdocs" directory. All files should be copied to the same sub-directory.

When installing on an Oracle Application Server on the same host as ORW, the commands one can use on a Unix Operating System are:

```
mkdir $ORACLE_HOME/Apache/Apache/htdocs/orw-eal
cd [ORW Install Home]/workspace/workspaceapp/ExternalAppLaunch/orw-eal
cp * $ORACLE_HOME/Apache/Apache/htdocs/orw-eal
```

To install these files on a different host, FTP the contents of the directory, [ORW Install Home] /workspace/workspaceapp/ExternalAppLaunch/orw-eal, to the remote system's HTTP Server's "htdocs" directory.

appinfo.xml

To reduce modifications to the JavaScript files, all application information used by the scripts is contained in the file, appinfo.xml. This file is located in the same directory as the supplied HTML and JavaScripts. The information is downloaded via AJAX (XMLHttpRequests) during the processing of the JavaScript.

This file contains the following elements:

- <osso-server>
This element is used to define where the OSSO Server is located. It contains the <scheme>, < host> and < port> elements. There must be only one <osso-server> element for each appinfo.xml file.
- < host>
This element contains the host name of the OSSO Server.
- < port>

This element contains the port number associated with the OSSO Server.

- `< scheme>`

This element specifies whether OSSO URLs should use http or https.

- `<application name= "appname" >`

This element is used to define additional information about an application not stored in OSSO. It may contain the following child elements: `<title>`, `<main-url>`, `<login-form-source>`, `<osso-id>`, `<credentials-submit-delay>`, and `<resubmit-delay>`.

- `<title>`

The text associated with this element is used for the HTML page title.

- `<main-url>`

This element holds the main application URL. See the specific HTML page description for more information.

- `<login-form-action>`

This element is used during the processing of the `jSecAppRemote.html` page to determine if the page just returned is the application's login page. The value of this element is a Regular Expression. A check is made against all forms' "action" attribute contained in the application's main page. If a match is not present, then the response is not a login page. If this element is missing or is empty, a match is assumed.

For `j_security_check` applications, the value of this element should be `"j_security_check"`.

- `<login-form-source>`

This element is used during the processing of the `jSecAppRemote.html` page to determine if the page just returned is the application's login page. The value of this element is a Regular Expression. A check is made against the Content-Location header to see if it matches this value. If a match is not present, then the response is not a login page. If this element is missing or is empty, a match is assumed.

- `<osso-id>`

This element holds the OSSO External Application ID. It is used during the processing of the `jSecAppRemote.html` and `jSecAppORW.html` pages.

- `<credentials-submit-delay>`

This element holds the number of milliseconds to delay before submitting a user's credentials to the external application. See the sections on `jSecAppRemote.html` and `jSecAppORW.html` for more details.

- `<resubmit-delay>`

This element is used during `jSecAppORW.html` process to specify the number of milliseconds to delay before re-requesting the main application URL. See the section on `jSecAppORW.html` for more details.

basicAppRemote.html

The HTML file `basicAppRemote.html` may be used when an external application uses BASIC authentication and users will be using recent versions of Internet Explorer (e.g. Internet Explorer version 7).

JavaScript executed during this page reads the username and password from the page's URL query string and converts these parameters into the appropriate HTTP "Authorization" header. The application URL is then requested via AJAX APIs and the page loaded into the current window.

To use basicAppRemote.html:

1. Install the contents of the orw-eal sub-directory into the same HTTP Server as the one servicing the External application.
2. Edit the appinfo.xml file and create an <application> element. Assign the "name" attribute with a unique value within the file. Assign a value to the <title> element. Put the URL of the application as the value of the <main-url> child element. A sample appinfo.xml file is seen below:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<app-info>
  <osso-server> <!-- Not used for basic authentication -->
    <host>myosso.mycompany.com</host>
    <port>7777</port>
    <scheme>http</scheme>
  </osso-server>
  <application name="myBasicApp" >
    <title>My BASIC Authenticated External Application Launcher</title>
    <main-url>
      <![CDATA[http://myhost.mycomputer.com:7781/myBasicApp/mainPage.
        html]]></main-url>
    </application>
  </app-info>
```

3. Define an External Application in OSSO.
 - a. Specify the "Login URL" to be the URL of the basicAppRemote.html file installed in step 1.
 - b. Specify the authentication type as "GET". Do NOT specify "BASIC".
 - c. Specify the "User Name/ID Field Name" as "username".
 - d. Specify the "Password Field Name" as "password".
 - e. Specify an "Additional Field" with the name of "app-name" and a value equal to the "name" attribute of the <application> element created in step 2.
 - f. Specify any additional fields needed by the application.
 - g. Obtain the OSSO ID of the application just defined. In the examples below, an ID of B9393469CEC51BDBBD9A8FA37DCDACB2 is used.
4. Configure retail-workspace-page-config.xml to launch the External Application via the OSSO External Application Launch URL. An example of this is seen below. In the example, the OSSO ID used is B9393469CEC51BDBBD9A8FA37DCDACB2. The OSSO ID of your External Application will be different.

```
<secure-work-item id="launchMyBasicExternalApp"
  display-string="My BASIC External Application" rendered="true"
  launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wvssso_app_admin.fapp_process_
  login</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>B9393469CEC51BDBBD9A8FA37DCDACB2</value>
    </parameter>
  </parameters>
```

```
</secure-work-item>
```

5. Configure Retail Workspace to allow a user to manage his/her credentials for that specific application. An example of this is seen below. In the example, the OSSO ID used is B9393469CEC51BDBBD9A8FA37DCDACB2. The OSSO ID of your External Application will be different.

```
<secure-work-item id="userCredForMyBasicExternalApp"
display-string="Manage Credentials for My Basic External Application"
rendered="true" launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wvssso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>B9393469CEC51BDBBD9A8FA37DCDACB2</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Possible issues with basicAppRemote.html:

- The application URL is loaded via the information returned in an AJAX request to retrieve the contents of the `appinfo.xml` file. Errors encountered reading this information may not be internationalized.
- All parameters used in launching the application are included in the query string segment of `basicAppRemote.html`'s URL. These parameters include the user name and password used to launch the application.
- This script does not have internationalized error messages.

jSecAppORW.html

The HTML file `jSecAppORWe.html` may be used when an external application uses the `j_security_check` authentication mechanism. The JavaScript invoked by this page does not examine any HTML data returned from the External Application. This script may also be used for applications deployed on HTTP Servers different than the one hosting this file. Hence, this script may be useful for more applications than `jSecAppRemote.html`.

JavaScript executed during this page downloads the `appinfo.xml` file. It then uses the query-string parameter, "app-name" to look up entries in the `appinfo.xml` file. The following elements are referenced:

- `<osso-server>` and the contained `<host>`, `<port>`, and `<scheme>` elements
- The appropriate `<application>` element whose "name" attribute matches the value of the "app-name" query string parameter. It also uses the contained `<title>`, `<main-url>`, `<osso-id>`, `<credentials-submit-delay>`, and `<resubmit-delay>` elements.

The executed JavaScript works as follows:

1. It submits a form to the URL specified in the `<main-url>` element.
2. Upon completion of loading this reference, the JavaScript then submits a form to the OSSO `fapp_process_login` URL. If the browser session is not logged in, this should result in the application logging into the application.
3. Upon completion of loading the OSSO `fapp_process_login` URL, the JavaScript then re-submits the form used in step 1). This is to handle the case where step 2) failed because the user is already logged in.

To use jSecAppORW.html:

1. Create a new External Application in OSSO.
 - a. Specify the "Login URL" to be the URL to submit the authentication information to. I.e. the action associated with the login form when submitting the login information.
 - b. Specify the authentication type as "POST".
 - c. Specify the "User Name/ID Field Name" as "j_username".
 - d. Specify the "Password Field Name" as "j_password".
 - e. Specify any "Additional Fields" needed for logging in.
 - f. Obtain the OSSO ID of the external application just defined. In the examples below, an ID of AE243F9B5B1E5A94C76CB936DEA6033 is used.
2. Install the contents of the orw-eal sub-directory into an HTTP Server. This may be the HTTP Server servicing ORW or the remote HTTP Server servicing the External Application.
3. Edit the appinfo.xml file and enter the correct values for the OSSO Server and the external application. A sample file is seen below:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<app-info>
  <osso-server>
    <host>myosso.mycompany.com</host>
    <port>7777</port>
    <scheme>http</scheme>
  </osso-server>
  <application name="myExternalApp" >
    <title>My External Application Launcher</title>
    <main-url>
      <![CDATA[http://myhost.mycomputer.com:7781/myExtApp/mainPage.
      html]]></main-url>
    <credentials-submit-delay>0</credentials-submit-delay>
    <resubmit-delay>500</resubmit-delay>
    <osso-id>6AE243F9B5B1E5A94C76CB936DEA6033</osso-id>
  </application>
</app-info>
```

In the example, a 0.5 second delay will occur before the final resubmission of the main application page.

4. Test the script outside of Retail Workspace.
5. Create a new entry in Retail Workspace to launch the application via the jSecAppORW.html URL. A sample entry is seen below:

```
<secure-work-item id="launchMyExternalApp"
display-string="My External Application" rendered="true" launchable="true"
show-in-content-area="false">
  <url>http://myhost.mycomputer.com:7778/orw-eal/jSecAppORW.html</url>
  <parameters>
    <parameter name="app-name">
      <value>myExternalApp</value>
    </parameter>
  </parameters>
</secure-work-item>
```

6. Configure Retail Workspace to allow a user to manage his/her credentials. An example of this is seen below. In the example, the OSSO ID used is 6AE243F9B5B1E5A94C76CB936DEA6033. The OSSO ID of your External Application will be different.

```
<secure-work-item id="userCredForMyExternalApp"
display-string="Manage Credentials for My External Application" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>6AE243F9B5B1E5A94C76CB936DEA6033</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Possible issues with jSecAppORW.html:

- **A user MUST define his/her credentials before this script will successfully launch the external application.** The user defines credentials via the entry in the retail-workspace-page-config.xml file configured in step 6.
- The launching process will always submit forms to the main application URL twice and the login URL once (after redirection from OSSO). This is an inefficient process, especially for applications taking a long time to load.
- When the External Application uses a different URL than the one used for jSecAppORW.html, a user's browser may need to be configured to accept third-party cookies. Internet Explorer appears to accept third-party cookies when the URL is in the "Local Intranet" zone.

jSecAppRemote.html

The HTML file, jSecAppRemote.html, may be used when an external application uses the j_security_check authentication mechanism. The JavaScript invoked is more efficient than jSecAppORW.html. Except in limited situations, this page must be installed on the HTTP Server serving the External Application. Otherwise, its use of AJAX APIs will violate Single Origin policies. This page has an advantage over the jSecAppORW.html page, in that users without defined credentials for an application will be prompted for their credentials by OSSO.

Note: The jSecAppORW.html page does not handle the condition where OSSO prompts for user credentials. As such, a user must pre-define their credentials before using that script. The jSecAppRemote.html will allow OSSO to prompt for user credentials.

JavaScript executed during this page downloads the appinfo.xml file. It then uses the query-string parameter, "app-name" to look up entries in the appinfo.xml file. The following elements are referenced:

- <osso-server> and the contained <host>, <port>, and <scheme> elements
- The appropriate <application> element whose "name" attribute matches the value of the "app-name" query string parameter. It also uses the contained <title>, <main-url>, <login-form-action >, <login-form-source >, <osso-id>, and <credentials-submit-delay>.

The executed JavaScript then works as follows:

1. It uses an AJAX API to request the application URL specified in the <main-url> entry of the `appinfo.xml` file.
2. It examines all forms in the response and compares the value of the "action" attribute to the <login-form-action> entry. Note: the <login-form-action> entry is interpreted to be a regular expression. For `j_security_check` applications, the login form typically has an action of "`j_security_check`".
3. It compares the "Content-Location" header of the response to the <login-form-source> entry. Note: the <login-form-source> entry is interpreted to be a regular expression.
4. If both steps 2) and 3) find matches, the response is interpreted as the browser session is not logged into the application. The browser is then forwarded to the OSSO External Application `fapp_process_login` URL, which uses the user's stored credentials to log into the application.
5. Else, the browser is assumed to have already logged in previously and the main application URL is loaded into the browser's window.

To use `jSecAppRemote.html`:

1. Create a new External Application in OSSO.
 - a. Specify the "Login URL" to be the URL to submit the authentication information to. I.e. the action associated with the login form when submitting the login information.
 - b. Specify the authentication type as "POST".
 - c. Specify the "User Name/ID Field Name" as "`j_username`".
 - d. Specify the "Password Field Name" as "`j_password`".
 - e. Specify any "Additional Fields" needed for logging in.
2. Install the contents of the `orw-eal` sub-directory into the same HTTP Server servicing the External Application.
3. Edit the `appinfo.xml` file and enter the correct values for the OSSO Server and the external application. A sample file is seen below:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<app-info>
  <osso-server>
    <host>myosso.mycompany.com</host>
    <port>7777</port>
    <scheme>http</scheme>
  </osso-server>
  <application name="myExternalApp" >
    <title>My External Application Launcher</title>
    <main-url>
      <![CDATA[http://myhost.mycomputer.com:7781/myExtApp/mainPage.
html]]></main-url>
    <login-form-action>j_security_check</login-form-action>
    <login-form-source>*.login.html.*</login-form-source>
    <credentials-submit-delay>300</credentials-submit-delay>
    <osso-id>6AE243F9B5B1E5A94C76CB936DEA6033</osso-id>
  </application>
</app-info>
```

In the example above, a 0.3 second delay will occur before the credentials are submitted to the remote application. Also, it assumes a page whose URL contains the string "`login.html`" will be returned if the browser session needs to login.

4. Test the script outside of Retail Workspace.
5. Create a new entry in retail-workspace-page-config.xml to launch the application via the jSecAppRemote.html URL. A sample entry is seen below:

```
<secure-work-item id="launchMyExternalApp"
display-string="My External Application" rendered="true" launchable="true"
show-in-content-area="false">
  <url>http://myhost.mycomputer.com:7778/orw-eal/jSecAppRemote.html</url>
  <parameters>
    <parameter name="app-name">
      <value>myExternalApp</value>
    </parameter>
  </parameters>
</secure-work-item>
```

6. Configure Retail Workspace to allow a user to manage his/her credentials. An example of this is seen below. In the example, the OSSO ID of the External Application is 6AE243F9B5B1E5A94C76CB936DEA6033. The OSSO ID of your External Application will be different.

```
<secure-work-item id="userCredForMyExternalApp"
display-string="Manage Credentials for My External Application" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wssso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>6AE243F9B5B1E5A94C76CB936DEA6033</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Possible issues with jSecAppRemote.html:

- The jSecAppRemote.html and associated JavaScript files must be installed on the same HTTP Server servicing the external application. Otherwise, the downloading of the main application page will violate the browser's Single Origin Policy.
- This script does not have internationalized error messages.

j_security_check Applications

Some applications require an application page be requested before responding with the login page. These applications maintain state information associated with the browser session and will only display the login page if the browser has not logged into the application. If the user has not logged in, then the login page is displayed and a form on the login page is submitted when the user supplies a user-id and password.

This document refers to these types of applications as "j_security_check" applications. One may use the OSSO external application facility to submit credentials, but additional logic is needed to submit the login credentials for a user. ORW supplies two sample HTML pages containing examples of this logic: jSecAppORW.html and jSecAppRemote.html. Descriptions of these files are found in the previous section. A major difference between the two files is that jSecAppORW.html may be installed on any available HTTP Server, while jSecAppRemote.html may only be securely run if installed on the same HTTP Server as the one serving the j_security_app application.

One example of a "j_security_app" is Central Office. A more detailed explanation of the process used to launch Central Office is seen in the next section.

Oracle Retail Central Office

The Oracle Retail Central Office application is an External Application using the `j_security_check` authentication mechanism. The sample `retail-workspace-page-config.xml` file contains an entry for this application. However, this entry does not contain a valid OSSO application ID. This section lists the steps needed to integrate Central Office with ORW using the sample `retail-workspace-page-config.xml` file.

This section details how to integrate Central Office with ORW via the following steps:

1. Creating the OSSO External Application definition.
2. Obtain the OSSO Application ID.
3. Installing the `jSecAppORW.html` or the `jSecAppRemote.html` pages and associated JavaScript files.
4. Edit `appinfo.xml`.
5. Edit `retail-workspace-page-config.xml`
6. Launching Central Office

Defining the Central Office External Application

To create the Central Office information in OSSO, access the "Manage External Applications" link found in the "Admin Tools" folder of the "Tools" worklist of the ORW navigation pane.

Determine the Login URL: Enter the appropriate information for submitting the credentials to the application. If you bring up the main page with the URL of, `http://myhost.mycompany.com:7777/centraloffice/`, then the Login URL is:

`http://myhost.mycompany.com:7777/centraloffice/j_security_check`

Other values:

- **Application Name:** This entry may be whatever you want it to be. However, a good value is "Central Office".
- **User Name/ID Field Name:** This entry must be "j_username".
- **Password Field Name:** This entry must be "j_password".
- **Type of authentication used:** This entry must be "POST".
- **Additional Fields:** A single additional field should be created with a Field Name of "login", a Field Value of "Login", and the Display to User unchecked.

An example of an OSSO External Application definition for Central Office is shown below:

Figure 4–6 OSSO External Application definition for Central Office

The screenshot shows a web browser window titled "Edit External Application - Central Office". The address bar shows a URL starting with "http://mspdv171.us.oracle.com:7777/jpi/orasso/DRASSO.wvssso_app_admin_administer_fo...". The page content includes:

- External Application Login:** A section with instructions and four input fields:
 - Application Name: Central Office
 - Login URL: http://myhost.mycompany.cor
 - User Name/ID Field Name: i_username
 - Password Field Name: i_password
- Authentication Method:** A section with instructions and a dropdown menu set to "POST".
- Additional Fields:** A table with columns "Field Name", "Field Value", and "Display to User". The first row is filled with "login", "Login", and a checkbox.

Obtain the OSSO Application ID

Once the OSSO External Application information has been saved, click on the "Edit" link for this application. Look at the browser's address bar and find the value of the `p_app_id` parameter. This will be a 32 hexadecimal digit value.

In the examples below, a value of 6AE243F9B5B1E5A94C76CB936DEA6033 is used for the OSSO application ID.

Installing the jSecAppORW.html or jSecAppRemote.html and Associated JavaScript Files

To install `jSecAppORW.html`, go to the "ExternalApplicationLaunch" directory found under the installation directory for the Oracle Retail application. Copy the `orw-eal` directory to an HTTP Server's `htdocs` directory or other directory where the HTTP Server can serve the `jSecAppORW.html` and associated JavaScripts. This may be the HTTP Server for Central Office or the HTTP Server for Retail Workspace.

To install `jSecAppRemote.html`, go to the "ExternalApplicationLaunch" directory found under the installation directory for the Oracle Retail application. On the HTTP Server for Central Office, copy the `orw-eal` directory to an appropriate directory.

Edit appinfo.xml

In the `orw-eal` directory just copied to the HTTP Server, find the `appinfo.xml` file and edit it. Use the appropriate values for the OSSO Server and port numbers, as well as the URL for the Central Office application. Use the CDATA convention for URLs or other strings that may contain special characters.

A sample file is shown below:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<app-info>
  <osso-server>
    <host>myosso.mycompany.com</host>
    <port>7777</port>
    <scheme>https</scheme>
  </osso-server>
  <application name="CentralOffice" >
    <title>My External Application Launcher</title>
    <main-url>
      <![CDATA[http://myhost.mycomputer.com:7781/centraloffice/]]></main-url>
    <login-form-action>j_security_check</login-form-source>
    <login-form-source></login-form-source>
    <credentials-submit-delay>0</credentials-submit-delay>
    <resubmit-delay>500</resubmit-delay>
    <osso-id>6AE243F9B5B1E5A94C76CB936DEA6033</osso-id>
  </application>
</app-info>
```

Not all entries in the example above are used by both jSecAppORW.html and jSecAppRemote.html. jSecAppORW.html does not use <login-form-action> and <login-form-source>. jSecAppRemote.html does not use <resubmit-delay>.

Edit retail-workspace-page-config.xml

The Retail Workspace configuration file will need to be edited for launching Central Office and for managing user credentials. This file is found in the directory, \$ORACLE_HOME/j2ee/[OC4J Instance]/RetailWorkspace, where [OC4J Instance] is the name of the OC4J instance where Retail Workspace is installed.

Note: The examples below use the <secure-work-item> element. Usage of this element requires permission grants for those roles authorized to access them. Use the <work-item> element if all users, including unauthenticated users, are allowed access.

Central Office Launch Link

To enable ORW to launch Central Office, look for the following entry in the sample retail-workspace-page-config.xml:

```
<secure-work-item id="centralOffice"
display-string="{confMgs.centralOfficeTitle}" rendered="true" launchable="true"
show-in-content-area="false">
<url>UnconfiguredAppURL.jsp</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>Unconfigured</value>
    </parameter>
  </parameters>
</secure-work-item>
```

There are two items that need to be changed in this entry. First, the <url> element value must contain the URL of the jSecAppORW.html file. Second, the parameter <value> element, must contain the "name" used in the appinfo.xml for the Central Office application. You should also verify the value of the "rendered" attribute is set to "true".

The example below shows an entry to launch Central Office from the same HTTP Server as ORW using jSecAppORW.html:

```
<secure-work-item id="centralOffice"
display-string="{confMsgs.centralOfficeTitle}" rendered="true" launchable="true"
show-in-content-area="false">
  <url>/orw-eal/jSecAppORW.html</url>
  <parameters>
    <parameter name="app-name" >
      <value>CentralOffice</value>
    </parameter>
  </parameters>
</secure-work-item>
```

The example below shows an entry to launch Central Office from the HTTP Server serving Central Office (coserver.mycompany.com, port 7777) using jSecAppRemote.html:

```
<secure-work-item id="centralOffice"
display-string="{confMsgs.centralOfficeTitle}" rendered="true" launchable="true"
show-in-content-area="false">
  <url><![CDATA[http://coserver.mycompany.com:7777/orw-eal/jSecAppRemote.
html]]></url>
  <parameters>
    <parameter name="app-name" >
      <value>CentralOffice</value>
    </parameter>
  </parameters>
</secure-work-item>
```

Central Office User Credentials Link

A user will also need to manage the credentials (user name and password) for logging into Central Office. A link for this must also exist in the retail-workspace-page-config.xml file.

Look for the following entry in sample retail-workspace-page-config.xml:

```
<secure-work-item id="manageCOUserCrdtl"
display-string="{confMsgs.centralOfficeUserCredTitle}" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value></value>
    </parameter>
  </parameters>
</secure-work-item>
```

Here, only the OSSO Application ID (obtained earlier) needs to be entered. You should also verify the value of the "rendered" attribute is set to "true".

An example of a completed entry is:

```
<secure-work-item id="manageCOUserCrdtl"
display-string="{confMsgs.centralOfficeUserCredTitle}" rendered="true"
launchable="true" show-in-content-area="false">
  <url>#{dasUrl.baseUrl}pls/orasso/ORASSO.wwsso_app_admin.edit_fappuser</url>
  <parameters>
    <parameter name="p_app_id" >
      <value>6AE243F9B5B1E5A94C76CB936DEA6033</value>
    </parameter>
```

```
</parameters>  
</secure-work-item>
```

Launching Central Office

Each user must create their own credentials before launching with `jSecAppORW.xml`. This is done using the work item created in the section above labeled, "Central Office User Credentials Link".

The `jSecAppRemote.html` page does not have this restriction, as it allows OSSO to prompt for these values if none are available.

This chapter describes security mechanisms found in Oracle Retail Workspace (ORW).

Overview

ORW uses two basic concepts of security.

- Authentication—ensuring a user is who the user claims to be.
- Authorization—allowing (or denying) a user specific capabilities.

ORW leverages standard J2EE application programming interfaces to verify a user has been authenticated and is authorized to access ORW resources. Authentication and authorization are configured within the Oracle Application Server and are managed by the OAS container. Authentication is dependent on the configured *Identity Store* which holds the user and role information. Authorization is dependent on the presence of *Permission Grants* held in a security policy. The policy may be held in a `system-jazn-data.xml` file or in an Oracle Internet Directory (OID) LDAP server.

There are multiple security mechanisms in place with ADF and the ORW implementation. The first involves standard J2EE Web application security. Standard container managed security involves the Java Authentication and Authorization Service (JAAS). Oracle's implementation of this service is known as JAZN.

For a distributed system such as ORW, authentication should be provided by a Single Sign-On (SSO) system. An SSO system requires a user to enter credentials (user ID and password) only once per HTTP session. ORW supports the Oracle Single Sign-On system and can be configured to use other SSO implementations.

In order to leverage any OSSO subsystem, the dashboard must be configured with specific entries in its deployment descriptors and the OAS configuration files.

Authorization for accessing specific JSPX pages is provided by the ADF infrastructure as well. ADF provides mechanisms to control fine-grained access to JSPX files and to resources found on these files, such as database connections. When correctly configured, the ADF infrastructure will limit access without an application developer writing any security specific code.

The final source of security is supplied by the ORW framework. This software verifies that a user only sees the work lists, external links, and/or dashboards appropriate for a user.

Deployment Descriptor Security Considerations

Elements within a J2EE application's deployment descriptors have a large impact on the security mechanisms that are in place and the resources that are protected. This section lists those elements and files leveraged by ORW.

The first deployment descriptor is the `web.xml` file. This is a standard J2EE file included with all web applications. There are a few requirements for this file:

- It must define the ADF Authentication servlet, including initialization parameters.
- It must define the URI of the ADF Authentication servlet.
- It must define a security constraint on the ADF authentication servlet if access is to be granted to non-authenticated users. Otherwise, the security constraint must be placed such that a user must be authenticated before accessing the application.
- It must declare all logical roles referenced by the ADF Authentication servlet.

Another file with security elements is the `orion-application.xml`, a deployment descriptor that is specific to the OAS application server. This file contains a `<jazn>` element specifying the authentication subsystem and security identity the application server will use. The entries in this file can control the interface with the SSO system and the ability for the application to invoke a Login Module for credential verification. This information can also be managed via the OAS Enterprise Manager application.

When deploying an application or dashboard using Oracle Single Sign-On, the `<jazn>` element within the `orion-application.xml` must specify a "provider" attribute whose value is "LDAP" (specifying the permission grants are stored in OID) and a `<jazn-web-app>` subelement with an "auth-method" attribute having a value of "SSO". An example is shown below:

```
<orion-application ... >
...
<jazn provider="LDAP" jaas-mode="doAsPrivileged" >
<jazn-web-app auth-method="SSO" />
</jazn>
...
</orion-application>
```

When deploying an application or dashboard using a login module, different entries must be made. The "provider" attribute must have a value of "XML", specifying the permission grants reside in the `system-jazn-data.xml` file. Additional properties are required as well. An example is seen below:

```
<orion-application ... >
<library ...>
...
<jazn provider="XML" jaas-mode="doAsPrivileged" >
<property name="role.mapping.dynamic" value="true" />
<property name="custom.loginmodule.provider" value="true" />
</jazn>
...
</orion-application>
```

Additional entries may also be required to the `<jazn>` element to support a non-OSSO single sign-on system.

Single Sign-On

ORW supports Oracle Single sign-On for providing user authentication. When OSSO is used, the Oracle HTTP Server used to front-end ORW must be registered with an instance of Oracle Single Sign-On server. OSSO is dependent on OID and all instances of ORW must also be deployed on application servers using OID as their Identity Store.

Because ORW uses container managed authentication, other SSO implementations can be configured. The compatibility of these SSOs depends on their support of the Oracle Application Server and how the SSO will integrate with the complete suite of Oracle Retail applications.

Roles and Groups

Access to ORW resources is granted based on a user's assigned roles. A user's role assignment is stored in the Identity Store associated with an OC4J instance. If a user is assigned multiple roles, then the user will have the capabilities found in the union of all of these roles.

Roles are roughly equivalent to 'Group' membership in an LDAP: a user will be assigned a role for every public group he/she is a member. In this document, the term 'role' and 'group' are used interchangeably.

Note: When using Oracle Internet Directory, only public groups are equated to roles. A public group has the 'Group visibility' or the 'public visibility' attribute set to 'true'.

In many Identity Stores, roles may contain other roles. As such, the contained role inherits any and all capabilities from the container role. For example, if a 'Merchant' role contains a 'Super Merchant' role, then the members of the 'Super Merchant' role will have all of the capabilities of the 'Merchant' role. They will also have the added capabilities of the 'Super Merchant' role.

Also, many Identity Stores allow roles to be contained in multiple roles. For example, another role, 'Supervisors', may contain the 'Super Merchant' role as well. In this case, the capabilities of the members of the 'Super Merchant' role would include all capabilities of the 'Merchant' role, all of the 'Supervisors' role, plus any specific capabilities of the 'Super Merchant' role.

A retailer may create and use site-specific roles with ORW. However, all of the site specific roles must be contained by or be members of the Retail_Workspace_Users role. Failure to do so will result in authentication errors when the user logs into ORW unless additional modifications are made to `orion-web.xml` deployment descriptor.

Permission Grants

ORW users are authorized to access a secured resource (work item or Dashboard) based on the presence of an appropriate 'Permission Grant'. These permission grants are stored in the OID LDAP server or in the `system-jazn-data.xml` file. The Oracle Application Server (OAS) retrieves these grants as needed.

Permission Grants are defined with the following attributes:

- Every Permission Grant has a type. Each type has a set of 'actions' that can be granted.

- Every Permission Grant has a 'name' which identifies what the target of the grant is. For example, the target of a 'Work Element Permission Grant' is a secured node on a Navigation panel work list tree.
- Every Permission Grant has one or more 'Grantees' identifying the role or other entity to whom the desired capabilities is granted. With ORW, there are three different types of Grantees: roles, users, and the ADF Anyone role.

Note: Granting a capability to the ADF Anyone role is equivalent to making that capability available to every user, including users that have not yet authenticated themselves (logged in).

ORW uses two types of Permission Grants: Work Element Permission Grants and ADF Region Permission Grants.

- Work Element Permission Grants control what is visible and accessible in the Navigation panel of ORW for a user. Users will only see those secured work items that they have been granted permission to access.

Work items may have a parent/child relationship. When such a relationship exists and the parent is a secure work item, then a user must have permission to view and access both the parent and child work item in order to view and access the child work item in the Navigation panel. The Permissions Management Tool automatically recognizes and displays secure Work Element Permission Grants associated with the current `retail-workspace-page-config.xml` configuration file.

- ADF Region Permission Grants are used by the underlying ADF framework. A secured dashboard or secured JSP page will require an ADF Permission grant for the 'view' action before a user can access it. ADF Region Permission grants can also grant the ability to customize (change default values) or personalize (change personal values) certain parameters, depending on how the page was constructed.

The ORW Permissions Management Tool does not recognize nor seek ADF Region Permission Grants for a work item unless the `<secure-work-item>` element in the `retail-workspace-page-config.xml` file contains a `<custom-attribute>` tag with a name attribute of "adf-permission-target". The value of this target must be the package qualified name of the ADF page definition file associated with the work item's JSPX file.

In summary, Work Element Permission Grants allow certain users to see and access a secured work item in the Navigation pane. ADF Region Permission Grants allow certain users to execute or view the JSP or JSPX page associated with the work item.

Managing Permission Grants via the Permissions Management Tool

ORW Permission Grants can be managed via the ORW 'Permissions Management' administrative tool. This tool is found in the 'Tools' work list of the navigation pane and manages permission grants for every secure work item and secure work list found in the `retail-workspace-page-config.xml` file.

When permission grants are stored in OID, the Permission Management Tool can create or delete both Work Element and ADF Region permission grants. However, when permission grants are stored in a `system-jazn-data.xml` file, the Permission Management Tool can only view grants. Furthermore, only the grants stored in the `system-jazn-data.xml` file associated with the OC4J hosting the ORW application

will be visible in the tool. Permission grants in other OC4J instances configured to use different `system-jazn-data.xml` files will not be visible.

In order to create or delete Permission Grants stored in OID, one must have the correct permissions within the Oracle Internet Directory LDAP server.

Typically, this means the user ID one logs in with is a member of the JAZNAdminGroup (cn=JAZNAdminGroup,cn=Groups,cn=JAZNContext,cn=Products,cn=OracleContext).

Managing permission grants stored in one or more `system-jazn-data.xml` files can be performed via a set of shell scripts supplied with ORW or by the standard JAZN admin tool supplied with Oracle Application Server.

LDAP Access

When an LDAP is used as the Identity Store, it is accessed by the OC4J container within the application server and by the ORW application.

Connections to the LDAP server are established for the following reasons, each with their own configuration entries:

- When an OID LDAP server is used, the OC4J container will establish a connection to the LDAP server. This connection is used to read the permission grants stored in OID and to retrieve the user and group information. The connection parameters are located in the `jazn.xml` configuration file.
- The ORW application will establish a connection to the LDAP to read additional user information. The connection will be based on entries in the `ldap-config.xml` configuration file. This information will be extracted whenever the `ldap-config.xml`'s `<identity-store>` element is set to "OID" or "v3LDAP". These entries are also used by the Permissions Management tool to manage Permission Grants stored within an OID server.
- When OID is not used, the OC4J container will establish a connection to the LDAP server to perform user authentication. This connection is configured in an LDAPLoginModule entry found within the `system-jazn-data.xml` configuration file.

The `ldap-config.xml` file contains the distinguished name to use for logging on to the LDAP. This user account must have the ability to search and read user information stored in the LDAP. When OID is used, ORW also executes a proxy operation so that subsequent LDAP operations use the identity (and privileges) of the user's OSSO account.

The password for LDAP connections created by ORW is stored encrypted in an Oracle Wallet. The location of this wallet is found in the `ldap-config.xml` configuration file.

The ORW application can be configured to run without accessing any LDAP. In this case, the `ldap-config.xml` file must contain an `<identity-store>` element with a value of "XML". This configuration is independent of the OC4J container's configuration.

Password Storage

All passwords used directly by the ORW application are stored in an encrypted form within an Oracle Wallet. These include passwords for logging into reports servers and for the ORW application to login to the OID LDAP Server.

External Servers and Web Services

Workspace may be configured to access an LDAP server and web services offered by BPEL, Oracle BI EE, and BIP servers. The protocol used for this may be either non-secure or secure. In a production environment, the secure protocol should be used so that credentials information is encrypted when transmitted over the network. For the LDAP server, this means accessing the server via the "ldaps" schema or protocol. For Web Services, this means specifying the "https" schema. Secure connections also require a different TCP port number than non-secure connections.

Integration Methods and Communication Flow

This chapter provides a functional overview of how ORW integrates with other applications, including other Oracle Retail applications.

Overview

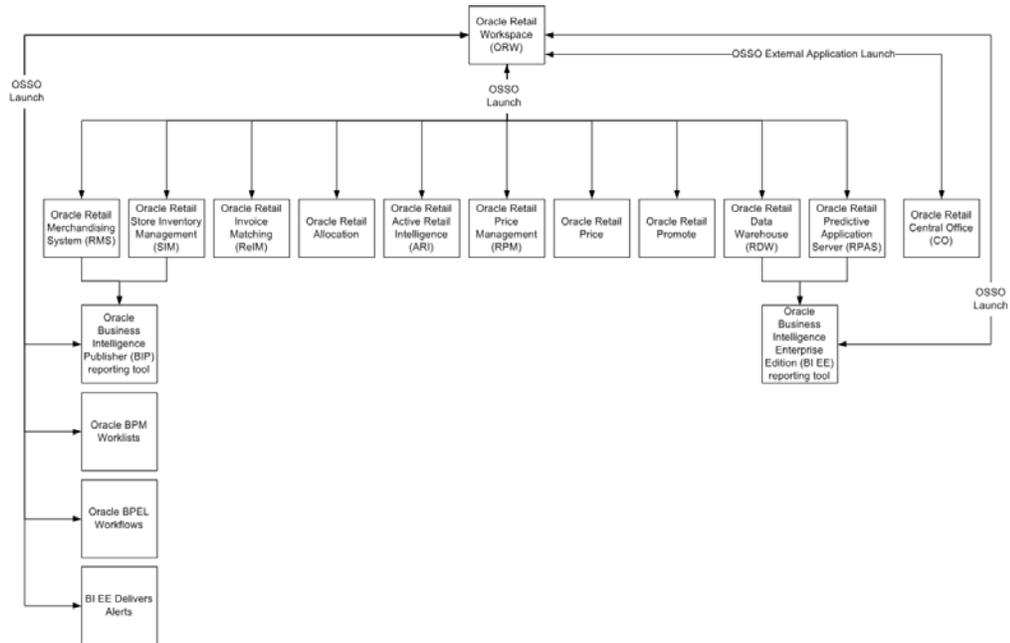
A diagram is provided illustrating the various Oracle Retail products ORW interfaces or launches.

System to System Integration

[Figure 6–1](#) details the overall integration among the various systems. The accompanying explanation of this diagram is written from a system-to-system perspective throughout the ORW-related portion of the enterprise. Note that this discussion focuses on a high-level functional use of data. For a technical description of the integration, see the sections later in this chapter.

Integration Interface Diagram

Figure 6–1 Integration Diagram



ORW and Oracle Retail Applications

ORW provides an easy to use, customizable, secure portal for launching applications and dashboards.

When launching one of the Oracle Retail applications that have been integrated into OSSO, the application leverages the same Oracle Internet Directory as the OSSO system. Hence, the same trusted authenticated user ID is used between ORW, OSSO, and the Oracle Retail applications. When launching an application as an External Application, such as Central Office, the application leverages the password mapping facility of OSSO.

Reporting Tools in ORW

ORW supports two reporting tools out of the box—Oracle BI Publisher (BIP) and Oracle Business Intelligence Enterprise Edition (BI EE).

When a user logs into ORW, each reporting tool is queried for a hierarchical list of reports that the logged in user can access. This list is then formatted and displayed in the Reports work list of the navigation pane.

BIP and Oracle BI EE reports may also be displayed in portlets on a dashboard.

Note: Due to the space constraints in portlets on a dashboard, reports displayed in these portlets may need to be resized. See the *Oracle Retail Reports Resizing Guide* which is available as My Oracle Support Doc ID 559554.1.

Oracle Retail Workspace and Single Sign-on

The SSO technology for ORW is Oracle Single Sign-On (OSSO). To process the login when an unauthenticated user accesses an OSSO protected application:

1. From a web browser, the client accesses the ORW URL through their Oracle HTTP server.
 - If the URL is statically protected, the browser is immediately redirected to the OSSO servlet.
 - If the URL is dynamically protected by the application logic, the browser is redirected to the OSSO servlet based on a specific HTTP response error.

In both cases, this redirect is performed by an Oracle Apache HTTP server `mod_osso` module.

2. The OSSO Servlet displays the OSSO login page. From here, the user enters a valid user name and password to access their defined role.
3. The user's login information is passed to the OID server and authenticated against the information defined in the OID server. The browser session is now considered authenticated.
4. The browser session loads the ORW main page.
5. When any linked URL is loaded, the OSSO sub-system validates that the browser session is authenticated and the page is displayed accordingly.

Integrating with Oracle Retail Applications

In order to access Oracle Retail applications through ORW, the `retail-workspace-page-config.xml` file must contain either the URLs or the External Application URLs of those applications. During installation of ORW, the installer queries for those URLs and modifies `retail-workspace-page-config.xml`. You will have to edit `retail-workspace-page-config.xml` if you add, move or delete applications. For additional information on setting up your application to use ORW, see the Operations Guide for the application or [Chapter 4, "External Applications"](#) of this guide.

Integrating with Report Servers

ORW supports two reporting tools out of the box: BI Publisher (BIP) and Oracle Business Intelligence Enterprise Edition (BI EE). The interface is optimal when those reports servers are OSSO enabled using the same Oracle Internet Directory (OID) as ORW.

There are two modes within ORW for interfacing with these reports servers: The first is exposing a hierarchical list of report hyperlinks in the Reports work list of the navigation panel. The second is showing reports within the portlets of a dashboard. These two modes are explained next.

Exposing Reports Links in the Navigation Panel

In order to expose BIP and Oracle BI EE reports links in the navigation panel, you must provide the necessary configuration during the ORW install. You can also edit the `<bip-reports-work-item>` and `<biee-reports-work-item>` XML elements in the `retail-portal-page-config.xml` file after installation. These elements are used

by ORW to query the reporting tools for all of the reports a user is allowed to access. They are also used to construct URLs to those reports. The names of reports are hierarchically displayed in the navigation panel as hyperlinks to those URLs. The hierarchy mimics that of the reporting tool. When a report's link is clicked, the report launches in the content area of the screen. If the report server is integrated with Oracle Single Sign-On, the report is launched without prompting for the user's credentials.

For more configuration detail see ["Customizing Reports Work Items"](#) in [Chapter 8, "Customization Guide"](#) of this guide.

Two custom attributes, the webservices login ID and the webservices URL prefix, are explained in detail below.

The Webservices Login ID

The <custom-attribute> elements named "bipublisher-login-id" and "biee-login-id" are used by ORW to make secure calls to BIP and Oracle BI EE webservices, respectively. These IDs must have administrative privileges in order to access reports on behalf of the logged-in user.

For BIP, the ID must be a BIP administrator.

For Oracle BI EE, the ID must be an Oracle BI EE administrator or as an impersonator.

See documentation for each of the BIP and Oracle BI EE products for information on how to set up these special users.

The Webservices URL Prefix

The <custom-attributes> elements named "bipublisher-webservices-url" and "biee-webservices-url" are used by ORW to make the calls to BIP and Oracle BI EE web services respectively. These web services return a list of reports the logged-in user is authorized to access.

In an OSSO environment, one may statically protect the main BIP and Oracle BI EE URLs in the mod_osso.conf file found in the HTTP Server configuration directory. If so, then the webservices URL prefix must be unprotected.

Unprotecting the BIP Webservices URL

For BIP, in a standard installation, the web.xml file located in the WEB-INF directory already contains a servlet mapping for BIP web services as shown below:

```
<servlet-mapping>
  <servlet-name>AxisServlet</servlet-name>
  <url-pattern>/services/*</url-pattern>
</servlet-mapping>
```

When the main BIP URL is protected, the mod_osso.conf file found in the HTTP Server configuration directory will have an entry as follows:

```
<Location /BIPublisher>
  require valid-user
  AuthType Basic
</Location>
```

An additional entry needs to be added to unprotect the BIP web services as follows:

```
<Location /BIPublisher/services>
  require valid-user
  AuthType Basic
  Allow from All
```

```
Satisfy any
</Location>
```

Unprotecting the Oracle BI EE Webservices URL

By default, the same URL header is used within Oracle BI EE to access all functions. When this URL is statically protected in the `mod_osso.conf` file (found in the HTTP Server configuration directory), the web services are therefore also protected. An example is as follows:

```
<Location /analytics/saw.dll >
    require valid-user
    AuthType Basic
    AuthName jazn.com
</Location>
```

To unprotect the Oracle BI EE web services, a separate URL reference for the Oracle BI EE services needs to be created.

In the `web.xml` file located in the `WEB-INF` directory of the Oracle BI EE installation, find the existing servlet mapping for the `SAWBridge` servlet shown below:

```
<servlet-mapping>
    <servlet-name>SAWBridge</servlet-name>
    <url-pattern>/saw.dll</url-pattern>
</servlet-mapping>
```

Then create another servlet mapping for that same servlet:

```
<servlet-mapping>
    <servlet-name>SAWBridge</servlet-name>
    <url-pattern>/services</url-pattern>
</servlet-mapping>
```

After this is done (and the Oracle BI EE application is re-started), access the web services in a browser using the URL:

```
http://<host>:<port>/analytics/services
```

If successful, use this URL as the `biee-webservices-url` value.

When accessing reports, ORW will use the `biee-reports-url-prefix` attribute. This URL should still be protected by OSSO and the example value is:

```
http://<host>:<port>/analytics/saw.dll
```

Showing Reports in Dashboards

ORW dashboards may reference the general purpose Report Portlet or URL Portlet deployed with ORW. These portlets use query parameters specifying external URLs to display. Such URLs may invoke BIP or Oracle BI EE reports. These URLs need to be carefully constructed and special characters properly escaped.

For information on how to construct BIP URLs refer to the *Creating Reports and Layouts* chapter of the *Oracle Business Intelligence Publisher User's Guide*.

For information on how to construct Oracle BI EE URLs refer to the *Integrating Oracle BI Presentation Services into Corporate Environments Using HTTP* chapter of the *Oracle Business Intelligence Presentation Services Administration Guide*.

A simple example BIP report's URL is:

```
http://<server>:<port>/BIPublisher/Guest/RMS/12.1dev/Orders/opo_merch_dash/opo_
```

merch_dash.xdo&_xmode=4

A more complex example BIP report's URL is:

```
http://<server>:<port>/BIPublisher/Guest/RMS/12.1dev/Orders/opo_merch_dash/opo_merch_dash.xdo?_xpf=&_xpt=0&_xdo=%2FGuest%2FRMS%2F12.1dev%2FOrders%2Fopo_merch_dash%2Fopo_merch_dash.xdo&_xt=OPO%20Merchant%20Dashboard%20Portal%20Report&_xf=html&_xmode=4
```

An example Oracle BI EE report's URL is:

```
http://<server>:<port>/analytics/saw.dll?Go&Path=%2Fshared%2FPortal%2FThis%20Week%27s%20Sales%20Contribution
```

An example Oracle BI EE dashboard's URL is:

```
http://<server>:<port>/analytics/saw.dll?PortalPages&PortalPath=%2Fshared%2FsalesTrend%2F_portal%2FsalesTrendDashboard
```

Some notable escape characters are:

- Space=%20 or +
- /=%2F
- Apostrophe=%27

Note: Due to the space constraints in portlets on a dashboard, reports displayed in these portlets may need to be resized. See the *Oracle Retail Reports Resizing Guide* which is available as My Oracle Support Doc ID 559554.1.

Integrating with Oracle BI Alerts

ORW provides two ways of integrating with Oracle BI Alerts:

- Exposing the Oracle BI Alerts Link in the Navigation Panel
- Displaying Oracle BI Alerts in an Alerts Portlet on a dashboard.

Oracle BI Alerts require an OSSO enabled Oracle BI EE server with iBots enabled for Delivers. For information on how to enable iBots for Delivers see Oracle BI EE documentation or My Oracle Support Doc ID 752517.1.

Exposing Oracle BI Alerts Link in the Navigation Panel

ORW provides the capability of launching Oracle BI Alerts generated by iBots in a separate window. In order to expose the Oracle BI EE Alerts link in the navigation panel, you can provide the necessary configuration during the ORW install. You can also edit the <secure-work-item> element identified by <secure-work-item id="biAlerts"> in the retail-portal-page-config.xml file after installation. You will need to set the rendered attribute to "true" and provide the Oracle BI EE Alerts URL that is of the form:

```
http://<host>:<port>/<analytics-context-root>/saw.dll?Alerts
```

where <host> and <port> are host and port numbers where Oracle BI EE presentation services are deployed and <analytics-context-root> is the context-root of Oracle BI EE presentation services.

Displaying Oracle BI Alerts in an Alerts Portlet on a Dashboard

ORW provides the capability of displaying Oracle BI EE Alerts generated by iBots, in a JSR-168 Portlet. The following describes the components and steps necessary to use this capability:

- An OSSO enabled Oracle BI EE server with iBots enabled for Delivers. For information on how to enable iBots for Delivers see Oracle BI EE documentaion or My Oracle Support Doc ID 752517.1
- An OBIEEAlerts Application (supplied with ORW) deployed on the same OAS server that hosts Oracle BI EE. This application's context-root must be OSSO protected statically.
- A deployment of ORWOBIEEAlertsPortlet, which is supplied with ORW in orw-portlets.zip.
- A dashboard page that consumes the ORWOBIEEAlertsPortlet.

Note: This dashboard page is not supplied with ORW. You will need to create this page if you intend to use ORWOBIEEAlertsPortlet. Refer to [Chapter 9, "Dashboard Development Tutorial"](#) of this guide for instructions on how to use and configure ORWOBIEEAlerts Portlet parameters to integrate it with Oracle BI EE Alerts.

Oracle BI EE provides the list of Alerts as a secured RSS Feed via a URL of format: `http://<host>:<port>/<analytics-context-root>/saw.dll?RssAlerts`. The OBIEEAlerts Application subscribes to the RSS Feed URL, parses the feed, and presents the list of active alerts in the `ORWOBIEEAlertsDisplay.jspx` page. The ORWOBIEEAlertsPortlet renders the content of the `ORWOBIEEAlertsDisplay.jspx` in an HTML IFRAME.

Due to the limitation imposed by browsers of not allowing access to cross-domain content, the OBIEEAlerts application is required to be deployed on the same OAS server hosting Oracle BI EE. To enable seamless SSO integration, the `ORWOBIEEAlertsDisplay.jspx` page is required to be statically protected. The following are the steps necessary to statically protect the `ORWOBIEEAlertsDisplay.jspx` page:

1. Ensure that the HTTP server for the Oracle Application Server has been registered with the Single Sign-On server and the `mod_osso` module enabled. For more information on how to register a partner application with SSO refer to the *Oracle Application Server Single Sign-On Administrator's Guide*.
2. Modify the `mod_osso.conf` of the HTTP Server of Oracle BI EE OAS to add the following lines:

```
<Location /ORWOBIEEAlerts>
    Header unset Pragma
    OsoSendCacheHeaders off
    AuthType Basic
    require valid-user
</Location>
```

where 'ORWOBIEEAlerts' is the context root where the application is deployed. If you changed the context-root during installation of the ORWOBIEEAlerts application, change 'ORWOBIEEAlerts' above to the context-root you specified during installation. If necessary, modify the `httpd.conf` file to include `mod_osso.conf` and restart the HTTP Server.

Integrating with BPEL Workflows and BPM Worklists

ORW provides the capability of displaying BPEL Workflows and BPM Worklists generated by BPEL Server by exposing them as JSR 168 portlets. This section describes the components and steps necessary to use this capability.

1. Install an advanced version of SOA Suite 10.1.3.1 with an installation type of J2EE Server, Web Server and SOA Suite, and then apply patch set of 10.1.3.4.
2. Enable OSSO for the middle tier applications in SOA Suite, namely 'orabpel' and 'hwservices'.
3. Configure the Identity Service in 'hwservices' application to use OID server for authentication.

The following steps explain the configuration process:

- a. Ensure that the ORACLE_HOME environment variable is set to the root directory of the Oracle Application Server of the SOA suite instance being configured.
- b. Open an operating system command prompt and go to the following directory, which includes the configuration scripts:


```
$ORACLE_HOME/bpel/system/services/install/ant-tasks
```
- c. Execute either `configure_oid.bat` (for Windows operating systems) or `configure_oid.sh` (for UNIX operating system) with the required parameters. Oracle recommends you use the bash shell to execute the script on Linux. For example, to run this script on Linux:

```
sh ./configure_oid.sh oid_admin_user oid_admin_passwd
oid_nonssl_port ssl_enabled oid_realm_name [seedAllUsers] |
[seedRequiredUsers]oc4j_admin_user oc4j_admin_passwd oc4j_container_name
```

For example:

```
sh ./configure_oid.sh orcladmin welcome 389 false us seedAllUsers oc4jadmin
welcome1 oc4j_soa
```

The execution of this command internally modifies the `$ORACLE_HOME/bpel/system/services/config/is_config.xml` file.

The file contents should look similar to the following:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<ISConfiguration
xmlns="http://www.oracle.com/pcbpel/identityservice/isconfig">
  <configurations>
    <configuration realmName="us" displayName="us Realm">
      <provider providerType="JAZN" name="OID">
        <connection url="ldap://my.oid.com:389"
          binddn="cn=orcladmin" password="passwd"
          encrypted="false"/>
      </provider>
    </configuration>
  </configurations>
</ISConfiguration>
```

The `configure_oid` script creates BPEL users and roles, and grants required privileges to the `BPMSysAdmin` role and the `BPMDefaultDomainAdmin` role.

- d. Ensure that the realms information in `wf_config.xml` and `is_config.xml` files located at `$ORACLE_HOME/bpel/system/services/config` folder is in sync.

For more detail on the configuration process refer to the Service Configuration section of the *Oracle BPEL Process Manager Administrator's Guide*.

4. Deploy the ORWBPELPortlets application (supplied with ORW) in its own OC4J instance and on the same OAS server that hosts the ORW application.
5. Deploy the ORWBPELPages application (supplied with ORW) in its own OC4J instance and on the same OAS server that hosts the ORW application.
6. Incorporate the BPEL roles into the ORW roles as explained below. This can be achieved by using the Oracle Identity Management Provisioning Console.
 - a. Add 'BPManalyst Role' and 'BPEL Process Manager Public Role' to Retail_Workspace_Users role.

Note: For the `bpeladmin` user to access data for BPM Worklists and BPEL Workflows, `bpeladmin` needs to be added to 'BPEL Process Manager Public Role'.

- b. Add 'BPMSystemAdmin Role' to 'DEMO_Workspace_Admin' role.
7. To integrate BPELWorkflows and BPMWorklist with ORW you must provide the necessary configuration during ORW install. The installer will prompt you for following URLs.
 - **BPEL Admin URL** - This URL is used to administer BPEL Workflows/Processes and Domains on the BPEL server.

This URL is exposed as a link in the navigation Panel of the ORW application.

The URL is of the form:

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

where `<host>` and `<port>` are host and port numbers where SOA Suite was installed

- **BPELWorkflow URL** - This URL is used to display the ORWBPELWorkflow page which consumes the ORWBPELWorkflowPortlet. This URL is exposed as a link in the navigation panel of the ORW application.

The URL is of the form:

```
http://<host>:<port>/<context-root>/faces/ORWBPELWorkflow.jspx
```

where `<host>` and `<port>` are host and port numbers of the server where the ORWBPELPages application was deployed and `<context-root>` is the context-root of the ORWBPELPages application.

- **BPMWorklist URL** - This URL is used to display the ORWBPMWorklist page, which consumes the ORWBPMWorklist Portlet. This URL is exposed as a BPMWorklist link in the navigation pane of the ORW application.

The URL is of the form:

```
http://<host>:<port>/<context-root>/faces/ORWBPMWorklist.jspx
```

where `<host>` and `<port>` are host and port numbers of the server where the ORWBPELPages application was deployed and `<context-root>` is the context-root of the ORWBPELPages application.

- ORWBPELPortlets Producer URL** - This URL is the portlets producer URL used in both ORWBPELWorkflow and ORWBPMWorklist pages. It is obtained at the end of the ORWBPELPortlets install.

The URL is of the form:

```
http://<host>:<port>/<context-root>/portlets/wsrp2?WSDL
```

where `<host>` and `<port>` are host and port numbers of the server where the ORWBPELPortlets application is installed and `<context-root>` is the context-root of the ORWBPELPortlets application.

You may also edit the `<secure-work-item>` element(s) identified by `id="bpmWorklists"` and/or `id="bpelWorkflows"` in the `retail-portal-page-config.xml` file after installation. You need to set the **rendered** attribute and **show-in-content-area** attribute of each of the respective work items to "true" if you want to display the work items in the content area.

Note: The BPMWorklist portlet allows run-time customization and personalization. ORW users can view and personalize the BPMWorklist page. ORW administrative users can view, customize, and personalize the BPMWorklist page.

To see the tasks in the BPMWorklist, the BPEL users must be assigned to the Retail_Workspace_User role. For more information, see Step 6.a above.

Configuring Role Based Access to BPEL Workflows

ORWBPELWorkflowPortlet has a mechanism to limit access to workflows based on the logged-in user role. To use this feature the deployment descriptor (`bpel.xml`) of the BPELProcess must declare the roles allowed to access the process in its `<configurations>` tag.

The following example configuration will allow only the users in the BPMAlyst role to access the VCWorkflow process.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BPELSuitcase>
  <BPELProcess id="VCWorkflow" src="VCWorkflow.bpel">
    <partnerLinkBindings>
      <partnerLinkBinding name="client">
        <property name="wsdlLocation">VCWorkflow.wsdl</property>
      </partnerLinkBinding>
      <partnerLinkBinding name="TaskService">
        <property name="wsdlLocation">TaskServiceWSIF.wsdl</property>
      </partnerLinkBinding>
    </partnerLinkBindings>
    <configurations>
      <property name="role" encryption="plaintext">BPMAlyst</property>
    </configurations>
  </BPELProcess>
</BPELSuitcase>
```

ORWBPELWorkflowPortlet also supports multi-role access. These roles must be separated by a ';' separator as shown below. The BPEL administrator can update these

roles at run time thru BPELAdmin console. For more details refer the *Oracle BPEL Process Manager Developer's Guide*.

```
<configurations>  
  <property name="role"  
encryption="plaintext">BPMAnalyst;BPMWorkflowAdmin</property>  
</configurations>
```

Functional Design and Overview

This chapter provides information concerning the various aspects of ORW functional areas.

Product Overview

ORW is a retail-specific next-generation portal providing an integrated, targeted user experience for interacting with Oracle Retail applications and data. ORW functionality provides the following features:

- Single sign-on
- Central launch
- Role based security
- Dashboard creation and viewing
- Integration with Oracle BI EE and BIP reports servers
- Integration with Oracle BI Delivers Alerts
- Integration with Oracle BPM Worklists
- Integration with Oracle BPEL Workflows
- User management
- Client specific customization

Along with all the above capabilities, ORW provides default roles and dashboards for demonstration purposes.

Single Sign-On

Single Sign-On (SSO) is the ability to access multiple applications using a single authentication to a central facility. The central SSO facility is responsible for securely authenticating the user and making this information accessible to the application. The application is then responsible for accessing the SSO authenticated user identity and behaving accordingly.

ORW is not the central SSO facility mentioned above. Like other SSO compliant applications, ORW is dependent on the central SSO facility for user authentication. Hence, the Single Sign-On capabilities of any application are independent of the presence of ORW. Users logging into the SSO system before accessing ORW will be recognized as such. The SSO technology ORW uses is Oracle Single Sign-On (OSSO).

There are several Oracle Retail applications that have been integrated into OSSO. In addition, an application that has not been integrated into OSSO can be configured as

an external application in OSSO. This ability allows such an external application's user credentials to be mapped for SSO when launched from ORW.

Central Launch

ORW provides a central point-of-access to launch applications, reporting tools, and reports. This central launch functionality uses OSSO for the Oracle Retail supported resources to pre-authenticate user security. This eliminates the need for the user to log in to each resource that is launched.

Role Based Security

ORW provides role based security functionality by utilizing OID DAS user management. Role based security allows resource access to be granted based on a role rather than an individual user. This applies to the accessibility of applications, dashboards, and user administrative tools from ORW. However, it does not apply to user authorization within those resources. User authorization for the application occurs separately from ORW and within the application's security management system.

Only OID DAS administrators can manage roles and associated permissions. It is recommended that an ORW administrator be designated as an OID DAS administrator. This allows the ORW administrator to manage roles and associated permissions. Role administration includes creating new roles and role hierarchies, editing existing roles, and assigning permissions to resources.

Role based security will manifest itself by displaying the permissible resources in the ORW navigation pane. All resources that are displayed are then available to the user for launching and viewing.

All users accessing ORW should belong to the Retail_Workspace_User role. Retailers may define their own specific roles, and may include them as members of the Retail_Workspace_User role.

Dashboard Creation and Viewing

ORW provides a development kit and detailed instructions for creating dashboards. Dashboards can contain reports, RSS feeds, HTTP links, HTML pages, Oracle BI Delivers Alerts, BPM Worklists, BPEL Workflows, any URL-based information, JSR 168 compliant portlets, and ADF components. The dashboards are viewable within ORW and role based security can be applied to them. See [Chapter 9, "Dashboard Development Tutorial"](#) for more details.

Reports

ORW integrates with the Oracle Business Intelligence Enterprise Edition (BI EE) and Business Intelligence Publisher (BIP) reports servers. ORW can be configured to show in the navigation panel a hierarchical list of Oracle BI EE and BIP reports and dashboards.

Oracle BI Delivers Alerts

ORW integrates with Oracle BI Delivers Alerts. This application can be launched from the navigation pane. It is also possible to display Oracle BI Delivers Alerts on a dashboard using the ORW Alerts portlet.

BPM Worklists

ORW integrates with BPM Worklists. This application can be launched from the navigation pane. It is also possible to display BPM Worklists on a dashboard using the ORW BPM Worklists portlet.

BPEL Workflows

ORW integrates with BPEL Workflows. This application can be launched from the navigation pane. It is also possible to display BPEL Workflows on a dashboard using the ORW BPEL Workflows portlet. ORW also provides a link to the BPEL administration tool to administer BPEL Workflows.

User Management

ORW provides links to OID Delegated Administrative Services. These screens are used by an administrator to manage role based security and by users to manage their profiles.

Client Specific Customization

ORW provides the ability to customize navigation work lists to display only relevant links and to organize them in an optimal manner. ORW also provides the ability to configure homepages, client branding, and look and feel.

Example Roles and Dashboards

Upon installation, ORW is configured with seven predefined roles. These roles are provided as examples and for administrative purposes. The examples include four business roles with predefined permissions to dashboards, applications, and management tools. The administrative roles are for managing roles and permissions. An **Anyone** role is part of the ADF framework and includes all users, both authenticated and unauthenticated.

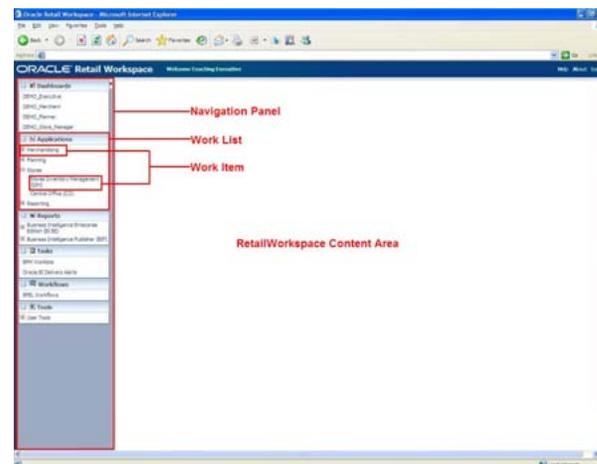
Upon installation, ORW includes four demonstration dashboards. These dashboards have predefined content that is relevant to an associated business role. The dashboard report content sources are ORW supported Oracle Retail applications. Therefore, the content of the demonstration dashboards is OSSO compliant.

Customization Guide

Customizing Oracle Retail Workspace Navigation

The Oracle Retail Workspace (ORW) application features a navigation panel that allows the user to navigate to various types of content, including dashboards, Oracle Retail applications, Oracle Business Intelligence Enterprise Edition (BI EE) and Business Intelligence Publisher (BIP) reports, Oracle BI Delivers Alerts, BPEL Workflows, BPM Worklists and other types of web pages.

Figure 8–1 Oracle Retail Workspace Navigation



The navigation panel is subdivided into one or more containers called **work lists**. A work list is used to group content of similar types. The default navigation panel has six work lists: Dashboards, Applications, Reports, Tasks, Workflows and Tools.

A work list contains a hierarchy of nodes called **work items**. Each work list has zero or more work items at the root level, and work items may have zero or more child work items. Work items that have children are similar to a directory (or folder) in a file system. The work item hierarchy may go several levels deep.

Work items that have children can be expanded and collapsed. Work items that do not have child work items, in other words "leaf" work items, usually have an action associated with them. For example, when a leaf node that represents an Oracle Retail application or a web page is clicked, the ORW application launches the application or

web page outside of the ORW window. Other leaf nodes, such as those that represent dashboard pages or reports, launch the content within the ORW content area.

Work lists and work items have several properties in common. For example, both may have child nodes, and in both cases the type of child node is a work item. Both may have display-string attributes, which define the string that is displayed in the UI for the work list or work item. Work lists and work items are sometimes collectively referred to as work elements.

The contents of the ORW navigation panel are defined in an XML configuration file. By default, the configuration file is named `retail-workspace-page-config.xml` and is installed in `<ORACLE_HOME>/j2ee/<instance name>/RetailWorkspace`, where `<ORACLE_HOME>` is the ORACLE_HOME environment variable for the application server where ORW is installed, and `<instance name>` is the name of the OC4J instance where ORW is installed.

After installing the ORW application, you may customize the contents of the navigation panel by editing `retail-workspace-page-config.xml`. You may edit the configuration to do any of the following tasks (and more):

- Change the titles of work lists
- Change the labels displayed for work items
- Change the URLs and query string parameters for work items
- Add new work lists and work items
- Remove work lists and work items
- Hide or un-hide work lists and work items
- Reorganize a work list's hierarchy
- Specify work items as role-based home pages

The following sections describe some of the more common customization tasks. For a more complete listing of ORW configuration parameters, refer to [Chapter 2, "Backend System Administration and Configuration"](#).

Changing the ORW Configuration File

To change `retail-workspace-page-config.xml`, you (or the person who administers or installs the ORW application) must have write access to the file and to the `<ORACLE_HOME>/j2ee/<instance name>/RetailWorkspace` directory. Changes to `retail-workspace-page-config.xml` affect all users of the ORW application.

Before you make changes to `retail-workspace-page-config.xml`, save a backup copy of its current contents. Then make a writable copy of the configuration file. If you encounter problems after making changes to the configuration, you can use the backup copy to restore the state of the configuration file.

Make changes to the copy of the configuration file as described in the various scenarios below, then save the file. After you finish making changes, copy the modified configuration file back to `<ORACLE_HOME>/j2ee/<instance name>/RetailWorkspace`, renaming to `retail-workspace-page-config.xml` if necessary. Changes take effect the next time users log in to ORW.

When configuring new values for elements in `retail-workspace-page-config.xml`, the values may include characters that you don't want the XML parser to parse (for example, `>`, `<`, `&`, `'`, and `"`). If a configuration value has characters that you want the XML parser to ignore, put your

text inside a CDATA section. A CDATA section starts with "<![CDATA[" and ends with "]]>". Everything inside the CDATA section is ignored by the parser. The ORW installer routinely encloses URL values in CDATA sections. Here is an example:

```
<parameter name="TOP_RSS_URL">
  <value><![CDATA[http://www.oracle.com/rss/rss_ocom_pr.xml]]></value>
</parameter>
```

When editing these values, edit the text between the beginning "<![CDATA[" and ending "]]>".

Internationalizing String Values in the Navigation Panel

The ORW configuration file supports internationalization and localization of string values. This feature allows you to customize your application in a manner that supports multiple language locales. You may define strings in a Java resource bundle, then configure a reference to the resource bundle in `retail-workspace-page-config.xml`. You then may reference properties from the resource bundle elsewhere in `retail-workspace-page-config.xml` using JSF Expression Language (EL) syntax.

Adding a reference to a custom resource bundle in `retail-workspace-page-config.xml` is as simple as locating the `<resource-bundles>` element, then adding another `<resource-bundle>` element inside this element.

Example: Creating and configuring a resource bundle

When customizing ORW, you may wish to replace the label of an element in the navigation panel with a new internationalized string. Let's assume you need a new string, "Last Week's Sales".

You should not modify the resource bundles that are supplied with the ORW application. Instead, create a new resource bundle. For this example, the resource bundle's base name is "CustomMessages", and it is created relative to the Java package name of "com.mycompany.bundles".

1. Create a file named `CustomMessages.properties` in a `com/mycompany/bundles` directory.
2. Edit `CustomMessages.properties` and define a resource bundle string whose default value is "Last Week's Sales":

```
lastWeeksSalesTitle=Last Week's Sales
```

3. Package and deploy the new resource bundle (and any translations) to the ORW application. Refer to ["Internationalization" in Chapter 2, "Backend System Administration and Configuration"](#) for more information about packaging and deploying resource bundles.
4. Add a reference to the new resource bundle to `retail-workspace-page-config.xml`. This is done by adding a new `<resource-bundle>` element to the `<resource-bundles>` element as in this example:

```
<resource-bundle var="customMsgs"
  resource-bundle="com.mycompany.bundles.CustomMessages"/>
```

When configuring a resource-bundle element, make sure that the value of the "var" attribute is unique and does not conflict with any other JSF variable in use by the ORW application.

By adding this resource-bundle element, you are able to reference values from the resource bundle by using EL expressions. In this example, we can refer to the "Last Week's Sales" value by referencing the following EL expression:

```
{customMsgs.lastWeeksSalesTitle}
```

Here is an example of how it might be used in `retail-workspace-page-config.xml`:

```
<parameter name="TOP_REPORT_TITLE">
  <value>#{customMsgs.lastWeeksSalesTitle}</value>
</parameter>
```

For more information about creating and deploying a custom resource bundle and internationalizing the ORW configuration, refer to ["Internationalization"](#) in [Chapter 2, "Backend System Administration and Configuration"](#).

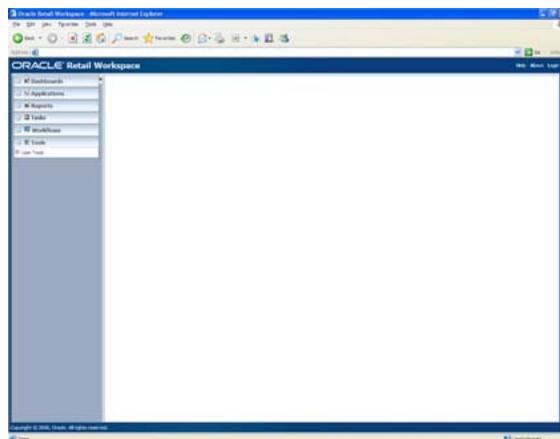
Securing Work Elements

Some of the scenarios in this chapter describe how to add or change work items and work lists. When you open `retail-workspace-page-config.xml`, notice that work items are configured using the `<work-item>` element or the `<secure-work-item>` XML element. Similarly, you can configure work lists by using `<work-list>` or `<secure-work-list>` elements. You can control the visibility and access to work items and work lists by defining them as "secure". When a work item or work list is secure, only users that belong to certain security roles (OID groups) may view and access the work item or list.

A work list is "unsecure" if it has been configured using the `<work-list>` element. Unsecure work lists may be viewed by any user. All work lists in the default ORW configuration are unsecure. Therefore, each of the default work lists may be viewed by all users unless the work list's "rendered" attribute has been set to "false". It's important to note that "unsecure" work lists can and do contain "secure" work items.

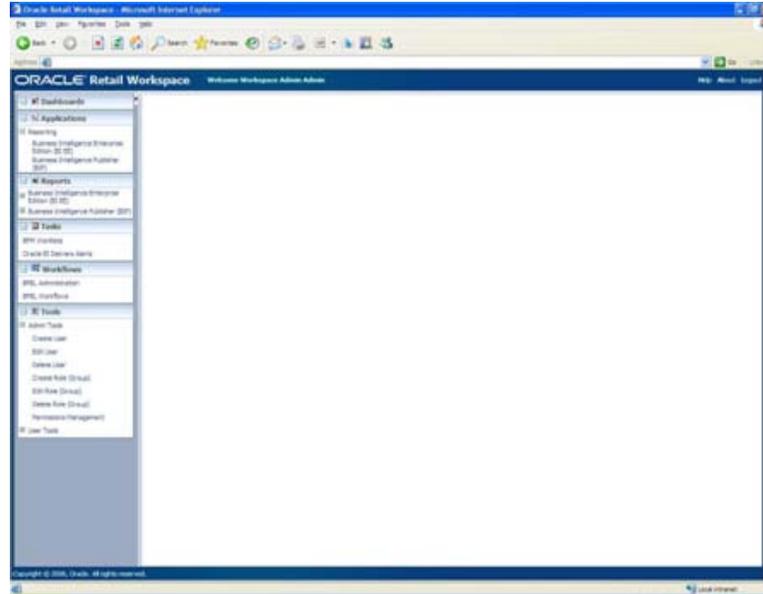
The following figure shows the ORW application before the user has logged into the application. Notice that all four default work lists are visible because they are "unsecure". Notice also that only the Reports and Tools work lists have visible work items. These are unsecure work items that may be viewed and accessed by any user.

Figure 8–2 Application Before the User Logs On



The following figure shows the same application after the "workspaceadmin" user has logged in. Notice that now reporting applications are available in the Applications work list, and that several administration tools are available in the Admin Tools folder of the Tools work list.

Figure 8–3 Application After the Workspaceadmin Logs On



In addition to configuring a secure work item or list using the <secure-work-item> or <secure-work-list> XML elements, you must also add or alter permission grants for the work item or list to grant permission to access the item or list.

Configuring Permission Grants When Using OID

When you add, change, or delete a secure work element, you may also need to add or delete permission grants associated with the work element. When ORW is configured to use OID as the LDAP server, the Permissions Management tool may be used to add or delete permission grants for a secure work element.

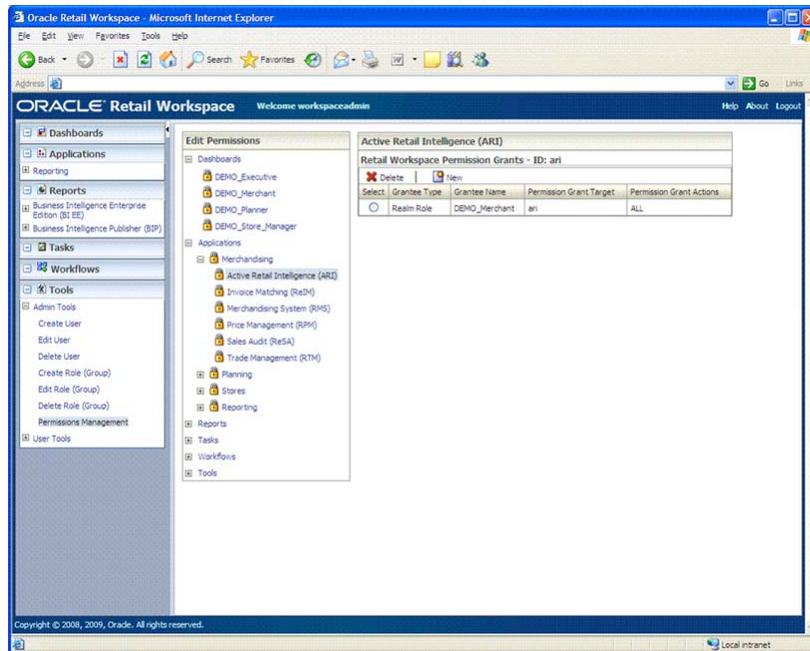
Note: The user ID must have specific privileges within OID to add new permission grants. Specifically, the user ID must be a member of the JAZNAdminGroup, found in OID with the distinguished name, "cn=JAZNAdminGroup,cn=Groups,cn=JAZNContext,cn=Products,cn=OracleContext"

Before a user can see a <secure-work-list> or a <secure-work-item> in the navigation panel, a permission grant must be created that allows the user access. Permission grants are typically made to roles (OID Groups) and not specific users. For <work-list> or <work-item> elements, no permission grants are needed to see the work element in the ORW navigation panel.

The following figure shows the Permissions Management screen. In this example, the Active Retail Intelligence (ARI) secure work item is selected. A permission grant exists for the DEMO_Merchant role. The result is that "Active Retail Intelligence (ARI)" appears in the "Merchandising" folder of the Applications work list for all users that belong to the DEMO_Merchant role. Of course this assumes that necessary

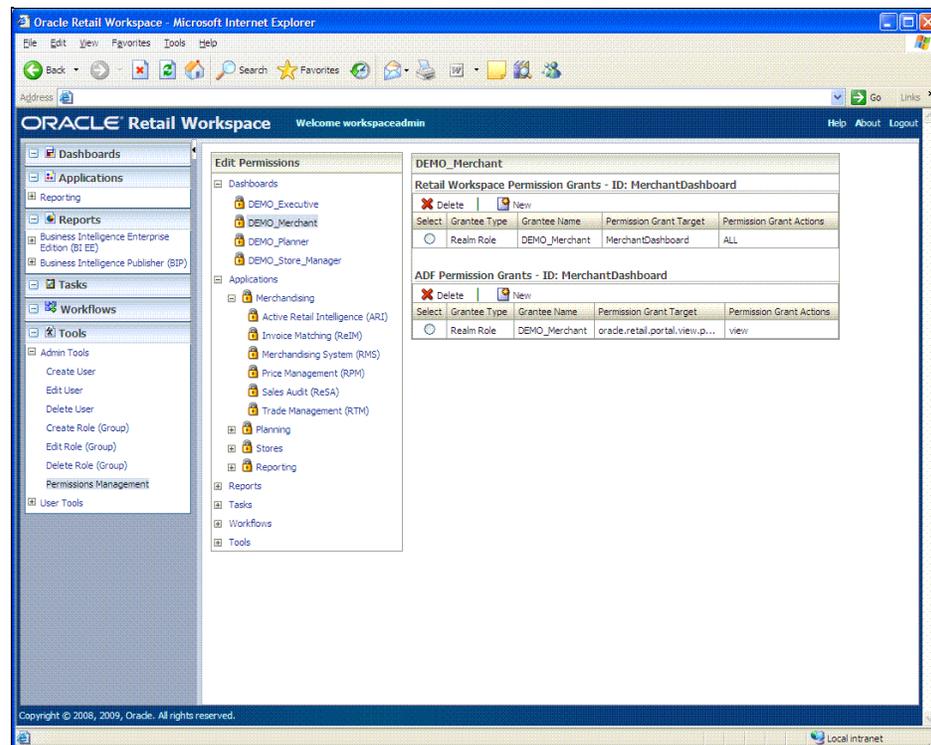
permissions have also been granted for the "Merchandising" folder to allow the user access to the folder.

Figure 8-4 Permissions Management Screen



Secure dashboard pages require an additional permissions grant to allow the user to actually view the dashboard. This permission is called an ADF Region Permission. ADF Region permission grants may be configured using the "ADF Permission Grants" list. The target of the ADF Permission Grant is the "page def" file of the dashboard page.

The following figure shows the DEMO_Merchant dashboard selected in the Permissions Management screen. An ORW Permission Grant exists for the DEMO_Merchant role. The result is that "DEMO_Merchant" appears in the Dashboards work list for all users that belong to the DEMO_Merchant role. An ADF Permission Grant also exists for the DEMO_Merchant role. This grant allows users that belong to the DEMO_Merchant role to actually view the dashboard page.

Figure 8–5 DEMO_Merchant Dashboard in the Permissions Management Screen

Note: When you use the Permissions Management tool to change permission grants, changes may not take effect immediately. This is because of application server caching. You may have to stop and restart the application server before the new permission grants take effect.

For additional information about the Permissions Management tool, refer to the *Oracle Retail Workspace Administration Guide*. Refer to the Dashboard Development Tutorial chapter for more details regarding securing a dashboard work item.

Configuring Permissions Grants to Allow Portlet Customization and Personalization

Secure dashboard pages that include portlets may be configured to allow run-time customization and personalization of the portlets that are displayed on the page. The following ORW portlets support run-time customization and personalization:

- URL Portlet
- Report Portlet
- RSS Portlet
- ORW Slideshow Portlet
- ORW BPM Worklist Portlet

Portlet customizations are default preferences that are visible by all users. Typically the Customize privilege is granted only to administrative users. Portlet personalizations are preferences that are visible only by the user that specified them. A user's

personalized preferences override the corresponding default customized preferences specified by the administrator. For more information on enabling customization and personalization of ORW portlets in a dashboard, see Chapter 9, "Dashboard Development Tutorial" for more information about enabling customization and personalization of ORW portlets in a dashboard.

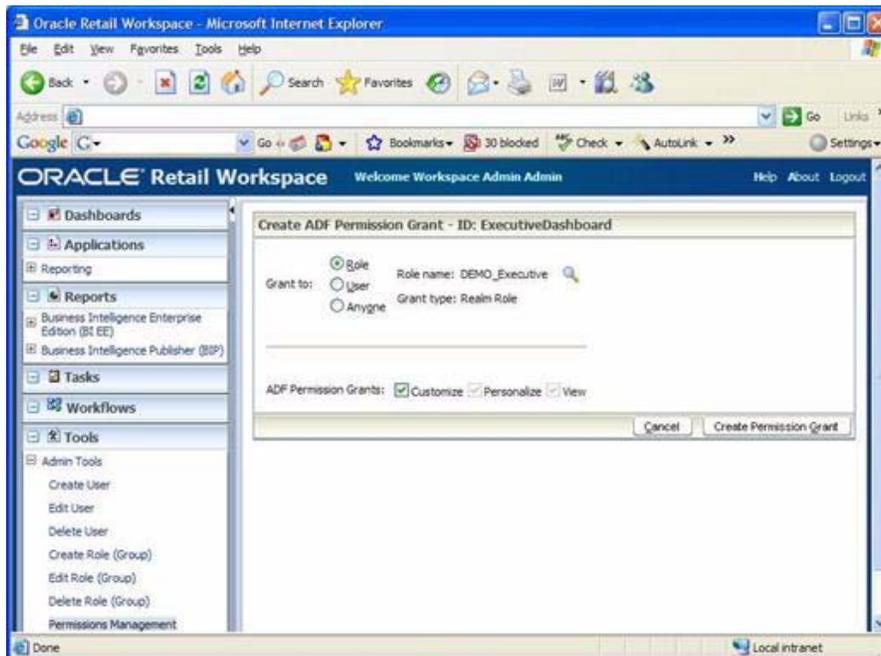
When ORW is configured to use Oracle Internet Directory as the LDAP server, privileges may be assigned by using the Permissions Management tool.

Customization and personalization privileges are granted via ADF Permission Grants, with one or more of the following actions:

- Customize
- Personalize

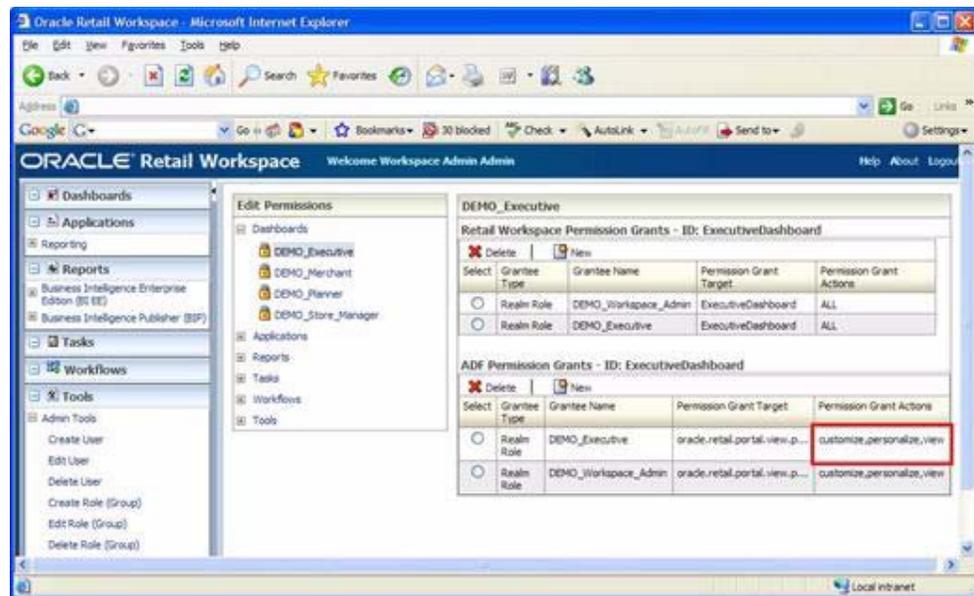
The following figure shows the Create ADF Permission Grant screen. In this example the administrator is about to grant Customize privileges for the ExecutiveDashboard work item for users in the DEMO_Executive role. Note that users with Customize privileges automatically get Personalize and View privileges.

Figure 8–6 Create ADF Permission Grant Screen



The following figure shows the Permissions Management screen after granting customize and personalize permissions to the DEMO_Executive role for the work item who's ID is equal to ExecutiveDashboard. Note that customize, personalize and view are listed as the permission grant actions.

Figure 8–7 Customize and Personalize Permissions Granted to DEMO_Executive Role



For more information on the Permissions Management tool, refer to the *Oracle Retail Workspace Administration Guide*.

Changing Existing Content in the Navigation Panel

During ORW application installation, the ORW installer configured and deployed ORW. The installer prompted for several parameters that affect the contents and behavior of the default Dashboards, Applications, and Reports work lists. This includes the URLs for Oracle Retail applications, the URLs and parameters for optional ORW example dashboards, the URLs and parameters for optional BI EE and BIP reporting tools, etc. The following sections describe how to change navigation panel configuration settings that were made by the ORW installer.

Changing a Work Item's URL and Query String Parameters

If the location of a dashboard, Oracle Retail application, or other content changes, you can change the work item's URL. It is also possible to change a work item's URL query string parameters.

Example: Changing the URL and parameters of a dashboard

The ORW application is packaged with four example dashboards. You entered the URLs to the deployed dashboard applications when installing the ORW application. You were also prompted to enter URLs to BI EE or BIP reports that the dashboards display.

In this example, we have decided to move the Demo Merchant Dashboard application to the host "prod.mycompany.com" from "dev.mycompany.com" and we now need to change the ORW configuration to point to the new location. We also have decided to change the top left report to display a custom report that shows last week's sales, and we will change the top report's title. We want the title to be internationalized.

1. Locate the work item that defines the dashboard in `retail-workspace-page-config.xml`. In this example, the work item is the

<secure-work-item> element with "id" attribute equal to "MerchantDashboard". It looks similar to the following fragment of the XML configuration:

```
<secure-work-item id="MerchantDashboard"
    display-string="{examplesMsgs.merchantDashboard}"
    rendered="true" launchable="true"
    show-in-content-area="true"
    target-frame="_iframe">

<url>http://dev.mycompany.com:7777/MerchantDashboard/faces/DemoMerchantDashboard.jspx</url>
    <parameters>
        <parameter name="TOP_REPORT_TITLE">
            <value>#{merchDashboardMsgs.topReportTitle}</value>
        </parameter>
        <parameter name="TOP_REPORT_URL_TO_SHOW">
<value><![CDATA[http://dev.mycompany.com:7777/analytics_
dev/saw.dll?PortalPages&PortalPath=%2Fshared%2FPortal%2F_
portal%2FResized-Top+Performers&nquser=administrator&nqpassword=rdw13]]></value>
        </parameter>
```

(parameters intentionally omitted)

2. Change the value of the <url> element. In our example, the dashboard has been deployed to

http://prod.mycompany.com:7779/MerchantDashboard/faces/DemoMerchantDashboard.jspx. To change the URL, simply replace the string between the <url> and </url> XML tags with the new URL:

```
<url>http://prod.mycompany.com:7779/MerchantDashboard/faces/DemoMerchantDashboard.jspx</url>
```

3. Change the TOP_REPORT_URL_TO_SHOW parameter to point to a new URL. In this example, the new report's URL is http://prod.mycompany.com:7777/analytics_dev/saw.dll?MyNewReport&PortalPath=%2Fshared%2FPortal%2FLast+Week+Sales&nquser=administrator&nqpassword=rdw13.

To change the TOP_REPORT_URL_TO_SHOW parameter, locate the dashboard's <parameter> element whose name is equal to TOP_REPORT_URL_TO_SHOW. Set the value of the <value> sub-element (i.e. replace the string between the <value> and </value>) to the new URL. In this case, because the URL contains characters like "&" that we don't want parsed by the XML parser, we need to wrap the value inside a CDATA section:

```
<parameter name="TOP_REPORT_URL_TO_SHOW">
<value><![CDATA[http://prod.mycompany.com:7777/analytics_
dev/saw.dll?MyNewReport&PortalPath=%2Fshared%2FPortal%2FLast+Week+Sales&nquser=
administrator&nqpassword=rdw13]]></value>
</parameter>
```

4. Define a resource bundle string whose default value is "Last Week's Sales". Refer to ["Internationalizing String Values in the Navigation Panel"](#) for more complete details about internationalizing strings.
5. Change the "TOP_REPORT_TITLE" parameter to specify the internationalized title string. To change "TOP_REPORT_TITLE", locate the dashboard's <parameter> element whose name is equal to "TOP_REPORT_TITLE". Set the value of the <value> sub-element to an EL expression that references the string that was just added to com.mycompany.bundles.CustomMessages:

```
<parameter name="TOP_REPORT_TITLE">
  <value>#{customMsgs.lastWeeksSalesTitle }</value>
</parameter>
```

Example: Changing the URL and parameters of an Oracle Retail application

The default ORW configuration includes links to several Oracle Retail applications in the applications work list. During installation, the ORW installer prompted you to select the Oracle Retail applications you want to link to from the ORW. The installer then prompted you to enter the URL for each application that you selected.

In this example, we have decided to move the Merchandising System application (RMS) to the host "prod.mycompany.com" from "dev.mycompany.com" and we now need to change the ORW configuration to point to the new location. We also need to change the value of the "config" parameter to "rms13prodhpsso".

1. Locate the work item that defines the application in `retail-workspace-page-config.xml`. In this example, the work item is the `<secure-work-item>` element with an "id" attribute equal to "rms". It looks similar to the following fragment of the XML configuration:

```
<secure-work-item id="rms"
  display-string="#{confMsgs.rmsTitle}"
  rendered="true"
  launchable="true"
  show-in-content-area="false">
  <url>http://dev.mycompany.com:7780/forms/frmservlet</url>
  <parameters>
    <parameter name="config">
      <value>rms13devhpsso</value>
    </parameter>
  </parameters>
</secure-work-item>
```

2. Change the value of the `<url>` element. In our example, the application has been deployed to `http://prod.mycompany.com:7781/forms/frmservlet`. To change the URL, simply replace the string between the `<url>` and `</url>` XML tags with the new URL:

```
<url>http://prod.mycompany.com:7781/forms/frmservlet</url>
```

3. Change the "config" parameter. In our example, we are changing it to "rms13prodhpsso".

To change the "config" parameter, locate the work item's `<parameter>` element whose name is equal to "config". Set the value of the `<value>` sub-element (i.e. replace the string between the `<value>` and `</value>`) to the new value:

```
<parameter name="config">
  <value>rms13prodhpsso</value>
</parameter>
```

While this example is for the RMS application's "config" parameter, the same basic principals apply for configuring any parameter element. Simply locate the work item's `<parameter>` element by name, then change the value of the `<value>` sub-element.

Changing a Work Item's Label

It is possible to change a work item's label. To change the label of any work item, you change it's "display-string" attribute.

The ORW application displays tooltips when a user hovers the mouse over a work item's label. If a work item does not have a "tooltip" attribute specified, the tooltip will default to the value of the "display-string" attribute.

Example: Changing the label of a work item

For this example, let's assume that you have decided to change the configuration of the Demo Planner dashboard work item to point to a custom planner dashboard. You have already configured the <url> and <parameters> elements and now wish to change the work item's "DEMO_Planner" label. You want to call it "Planner Dashboard", and you want the string to be internationalized.

1. Define the label in the resource bundle. In this example, we will just add it to a "CustomMessages.properties" that we have already created and configured in a <resource-bundle> element with "var" equal to "customMsgs". Here is the property string we will add to the resource bundle:

```
myDashboardTitle=Planner Dashboard
```

2. Change the value of the work item's "display-string" attribute. First locate the planner dashboard's work item in the configuration. Before making changes, the beginning of the element will look similar to the following fragment of XML:

```
<secure-work-item id="PlannerDashboard"
    display-string="#{examplesMsgs.plannerDashboard}"
    rendered="true" launchable="true"
    show-in-content-area="true"
    target-frame="_iframe">
    <url>http://...
```

Change the value of "display-string" to #{customMsgs.myDashboardTitle}:

```
<secure-work-item id="PlannerDashboard"
    display-string="#{customMsgs.myDashboardTitle}"
```

Example: Changing/setting the tooltip of a work item

The default `retail-workspace-page-config.xml` does not define tooltips for work items. The tooltip will default to the value of the "display-string" attribute if it is not set. It is possible to configure the tooltip for a work item. For this example, let's assume that you want to specify a tooltip for the "Business Intelligence Enterprise Edition (BI EE)" work item in the Reports work list. You want to change the tooltip to "BI EE Reports" and you want the tooltip internationalized.

1. Define the tooltip in the resource bundle. In this example, you will just add it to a "CustomMessages.properties" that you have already created and configured in a <resource-bundle> element with "var" equal to "customMsgs". Here is the property string you will add to the resource bundle:

```
bieeReportsTooltip=BI EE Reports
```

2. Add a "tooltip" attribute to the work item. First locate the work item in the configuration. Before making changes the work item will look similar to the following fragment of XML:

```
<biee-reports-work-item id="bieereports"
    display-string="#{confMsgs.bieeAppTitle}"
    rendered="true"
    launchable="false">
    ... (remainder of element
intentionally omitted)
```

The "tooltip" attribute is supported by all variants of work item elements, including:

- work-item
- secure-work-item
- biee-reports-work-item
- bip-reports-work-item

The "tooltip" attribute is also supported by all variants of work list elements.

Since "tooltip" is an attribute of the work item element, it is configured by inserting "tooltip=replace with your text value" in the work item element. In this example, just insert the tooltip attribute after the "id" attribute of the biee-reports-work-item element:

```
<biee-reports-work-item id="bieereports"
    tooltip="#{customMsgs.bieeReportsTooltip}"
    display-string="#{confMsgs.bieeAppTitle}"
    rendered="true"
    launchable="false">
```

Changing a Work List Title

It is possible to change the title of a work list. To change the title of any work list, change its display-string attribute.

The ORW application displays tooltips when a user hovers the mouse over a work list title. If a work list does not have a tooltip attribute specified, the tooltip defaults to the value of the display-string attribute.

The steps for changing the work list display-string and tooltip attributes are similar to the steps for changing a work item display-string and tooltip.

The following fragment shows the Dashboard work list before changing the display string and adding a tooltip:

```
<work-list id="DashboardWorkList"
    display-string="#{confMsgs.dashboardsWorklist}">
```

Below is the work list after changing the title to an internationalized string and adding a tooltip:

```
<work-list id="DashboardWorkList"
    tooltip="#{customMsgs.dashboardTooltip}"
    display-string="#{customMsgs.customDashboardsWorklistTitle}">
```

Hiding a Work List

It is possible to hide a work list globally by changing the value of the "rendered" attribute to false. A work list that is hidden is still configured in `retail-workspace-page-config.xml`, but is not visible in the navigation panel.

You also may restrict the visibility of a work list to users that belong to specific security roles. Refer to ["Changing an Unsecure Work List to a Secure Work List"](#) for information about configuring a work list to restrict access to users in a specific role or roles.

Example: Changing the "rendered" attribute to hide a work list

In this example, let's assume that you do not want the Tools work list displayed in the navigation panel, but you do not want to remove it from the configuration.

1. Locate the Tools work list in `retail-workspace-page-config.xml`. The beginning of the work list will look similar to the following fragment of XML:

```
<work-list id="ToolsWorklist"
  display-string="#{confMsgs.toolsWorklist}">
  <icon-uri>#{confMsgs.toolsWorkListIcon}</icon-uri>
  <work-items>
    <secure-work-item id="AdminToolsFolder"
      display-string="#{confMsgs.adminToolsFolder}"
      rendered="true">
```

2. Change the value of the "rendered" attribute to "false". If the "rendered" attribute is not configured for the work list, its default value is "true". In that case simply add `rendered="false"` to the work-list element:

```
<work-list id="ToolsWorklist"
  rendered="false"
  display-string="#{confMsgs.toolsWorklist}">
  <icon-uri>#{confMsgs.toolsWorkListIcon}</icon-uri>
  <work-items>
    <secure-work-item id="AdminToolsFolder"
      display-string="#{confMsgs.adminToolsFolder}"
      rendered="true">
```

Making a Hidden Work List Visible

A work list is hidden if the rendered attribute is set to false. It is possible to make a hidden work list visible by changing the value of the rendered attribute to true. If the work list is secure, its visibility will also be affected by permission grants that restrict access to the work list.

Hiding a Work Item

It is possible to hide a work item globally by changing the value of the "rendered" attribute to false. You also may restrict the visibility of a work item to users that belong to specific security roles. A work item that is hidden is still configured in `retail-workspace-page-config.xml`, but is not visible in the navigation panel.

Example: Changing the "rendered" attribute to hide a work item

In this example, let's assume that you do not want the Metalink work item displayed in the navigation panel, but you do not want to remove it from the configuration.

1. Locate the Metalink work item in `retail-workspace-page-config.xml`. The work item will look similar to the following fragment of XML:

```
<work-item id="userMetalink"
  display-string="#{confMsgs.metalinkTitle}"
  rendered="true"
  launchable="true"
  show-in-content-area="false" >
  <url>http://metalink.oracle.com</url>
</work-item>
```

2. Change the value of the "rendered" attribute from "true" to "false":

```
<work-item id="userMetalink"
```

```

        display-string="#{confMsgs.metalinkTitle}"
        rendered="false"
        launchable="true"
        show-in-content-area="false" >
        <url>http://metalink.oracle.com</url>
    </work-item>

```

Making a Hidden Work Item Visible

A work item is hidden if the "rendered" attribute is set to "false". It is possible to make a hidden work item visible by changing the value of the "rendered" attribute to "true". If the work item is secure, its visibility will also be affected by permission grants that restrict access to the work item.

Changing an Insecure Work List to a Secure Work List

You may make an insecure work list secure by using the <secure-work-list> element to define the work list. Once you have changed a <work-list> element to a <secure-work-list> element, you must use the Permissions Management tool to assign permission grants to the roles that are allowed to view the work list.

Example: Changing an insecure work list to a secure work list

In this example, you will change the Applications work list to a secure work list. By default the Applications work list is always visible. You wish to change the configuration to make the work list visible only when an ORW user has logged in. All of your ORW users are members of the OID "Retail_Workspace_Users" group.

1. Change the Tools work list to a <secure-work-list>. Locate the Tools work list in retail-workspace-page-config.xml. The work list begins with:

```
<work-list id="ApplicationsWorklist"
```

and ends with:

```
</work-list>
```

Simply replace work-list with secure-work-list. After making the change, the element will begin with:

```
<secure-work-list id="ApplicationsWorklist"
```

and will end with:

```
</secure-work-list>
```

After making this change the work list is secure. Since no permission grants have been configured for this work list yet, it will not be visible in the navigation panel.

2. Log in to the ORW application as a user that has permission to access the Permissions Management tool.
3. Open the Permissions Management tool.
4. Select the Applications work list node and use the Permissions Management tool to add an ORW permission grant for the Retail_Workspace_Users role.

Changing an Insecure Work Item to a Secure Work Item

You may make an insecure work item secure by using the <secure-work-item> element to define the work item. Once you have changed a <work-item> element to a

<secure-work-item> element, you must use the Permissions Management tool to assign permission grants to the roles that are allowed to access the work item.

Example: Changing an unsecure work item to a secure work item

In this example, you will change the User Tools folder of the Tools work list to a secure work item. By default the User Tools work item is always visible. You wish to change the configuration to make the work item visible only when an ORW user has logged in. All of your ORW users are members of the OID "Retail_Workspace_Users" group.

1. Change the User Tools folder work item to a <secure-work-item>. Locate the User Tools folder work item in `retail-workspace-page-config.xml`. The work item begins with:

```
<work-item id="UserToolsFolder"
```

and ends with:

```
</work-item>
```

Simply replace `work-item` with `secure-work-item`. After making the change, the element will begin with:

```
<secure-work-item id="UserToolsFolder"
```

and will end with:

```
</secure-work-item>
```

After making this change the work item is secure. Since no permission grants have been configured yet for this work item, the folder will not be visible in the navigation panel, even after a user has logged in.

2. Log in to the ORW application as a user that has permission to access the Permissions Management tool.
3. Open the Permissions Management tool.
4. Select the User Tools folder work item and use the Permissions Management tool to add an ORW permission grant for the `Retail_Workspace_Users` role.

Note: The user ID must also have permission within OID to create a permission grant. Specifically, the user ID must be a member of the `JAZNAdminGroup`, found in OID with the distinguished name, `"cn=JAZNAdminGroup,cn=Groups,cn=JAZNContext,cn=Products,cn=OracleContext"`.

Changing the ID of a Work Item or Work List

Each work item and work list has an "id" attribute that uniquely identifies the element. ORW uses the "id" attribute for two purposes:

- The work item ID is used when configuring the list of <home> work items. The "value" attribute of the <home> element must refer to the "id" attribute of a work item that is configured elsewhere in `retail-workspace-page-config.xml`.
- The "id" attribute of secure work items and secure work lists is required to be able to assign ORW permission grants.

If you are changing the ID of a work list or work item, you must make sure that you have removed all permission grants associated with the current ID. If you intend to

use the Permissions Management tool to remove permission grants, you must use it before you have changed the ID in the `retail-workspace-page-config.xml`. After you change the ID, you must make sure that any `<home>` element that refers to the old ID is either removed or changed to refer to the new ID. If the work list or work item is secure, you must also add back permission grants after changing the ID.

Example: Changing a Dashboard's ID

For this example, let's assume that you have decided to change the configuration of the Demo Planner dashboard work item to point to a custom planner dashboard with a new ID. You have already re-configured the work item without changing the ID, and have successfully tested the configuration. You now want to change the value of the "id" attribute to "MyCompanyPlannerDashboard".

1. In the ORW application, log in as a user that has permission to execute the Permissions Management tool. Navigate to the Permissions Management tool in the navigation panel and select it. Remove the "PlannerDashboard" permission grants one by one. Refer to the *Oracle Retail Workspace Administration Guide* for more information regarding the Permissions Management tool.
2. Change the value of the work item's "id" attribute to "MyCompanyPlannerDashboard". First locate the planner dashboard's work item in `retail-workspace-page-config.xml`. Before making changes, the beginning of the element will look similar to the following fragment of XML:

```
<secure-work-item id="PlannerDashboard" ...
```

Change the value of "id" to MyCompanyPlannerDashboard:

```
<secure-work-item id="MyCompanyPlannerDashboard" ...
```

3. Change any `<home>` element that references "PlannerDashboard". In this example, we want MyPlannerDashboard to serve as the home work item for the `demo_planner` role, so change the "value" attribute to reference MyCompanyPlannerDashboard:

```
<homes>
  <!--
    Each 'home' refers to the ID of the work item that
    is the default 'home page' for the users in the specified
    role. If a user is in multiple roles, the first matching
    role found is used. Role names must match the names of
    configured security roles.
  -->
  <home role="demo_executive" value="ExecutiveDashboard"/>
  <home role="demo_merchant" value="MerchantDashboard"/>
  <home role="demo_planner" value="MyCompanyPlannerDashboard"/>
  <home role="demo_store_manager" value="StoresDashboard"/>
</homes>
```

4. Save the changes to `retail-workspace-page-config.xml`. In the ORW application, log in as a user that has permission to execute the Permissions Management tool. In the Permissions Management tool, select the custom Planner Dashboard node, then add ORW permission grants for the DEMO_Planner role.

Customizing Reports Work Items

The default ORW navigation panel includes a Reports work list. The default configuration for the Reports work list has two specialized work items, one for Oracle Business Intelligence Enterprise Edition (BI EE) and the other for Business Intelligence Publisher reports (BIP):

- The Oracle BI EE Reports work item (configured in a `biee-reports-work-item` XML element)
- The BIP Reports work item (configured in a `bip-reports-work-item` XML element)

The Reports work items appear as folders in the Reports work list. Each of the Reports work items queries the reports server that is associated with it, and dynamically discovers the list of reports to display in the folder.

Figure 8–8 Reports Worklist



During installation, the ORW Installer prompted for various settings for Oracle BI EE and BIP. You may change the settings in `retail-workspace-page-config.xml` after installation. Oracle BI EE and BIP settings are defined in either URL query string parameters using the work item's `<parameter>` element, or in the `<custom-attributes>` element. Both `biee-reports-work-item` and `bip-reports-work-item` elements support custom attributes.

The following fragment of XML from `retail-workspace-page-config.xml` is an example of how a typical `biee-reports-work-item` is configured:

```
<biee-reports-work-item id="bieereports"
    display-string="#{confMsgs.bieeAppTitle}"
    rendered="true"
    launchable="false">
  <custom-attributes>
    <custom-attribute name="biee-webservices-url">
      <value>http://mycompany.com:7777/analytics/services</value>
    </custom-attribute>
    <custom-attribute name="biee-reports-url-prefix">
      <value>http://mycompany.com:7777/analytics/saw.dll</value>
    </custom-attribute>
    <custom-attribute name="biee-logon-id">
      <value>administrator</value>
    </custom-attribute>
    <custom-attribute name="biee-password-wallet-location">
      <value>/u00/webadmin/product/10.1.3_WC/OracleAS_1/
        j2ee/RetailPortal/RetailWorkspace/wallet</value>
    </custom-attribute>
  </custom-attributes>
</biee-reports-work-item>
```

```

</custom-attribute>
<custom-attribute name="biee-password-alias">
  <value>bieePwdAlias</value>
</custom-attribute>
<custom-attribute name="biee-shared-reports-folders">
  <value>shared</value>
</custom-attribute>
<custom-attribute name="biee-users-folder">
  <value>users</value>
</custom-attribute>
</custom-attributes>
<parameters>
  <parameter name="options">
    <value>rfd</value>
  </parameter>
</parameters>
</biee-reports-work-item>

```

The following fragment of XML from `retail-workspace-page-config.xml` is an example of how a typical `bip-reports-work-item` is configured:

```

<bip-reports-work-item id="bipreports"
  display-string="#{confMsgs.bipAppTitle}"
  rendered="true"
  launchable="false">
  <custom-attributes>
    <custom-attribute name="bipublisher-webservices-url">
      <value>http://mycompany.com:7777/xmlpserver/services</value>
    </custom-attribute>
    <custom-attribute name="bipublisher-reports-url-prefix">
      <value>http:// mycompany.com:7777/xmlpserver</value>
    </custom-attribute>
    <custom-attribute name="bipublisher-logon-id">
      <value>admin</value>
    </custom-attribute>
    <custom-attribute name="bipublisher-password-wallet-location">
      <value>/u00/webadmin/product/10.1.3_WC/OracleAS_1/
        j2ee/RetailPortal/RetailWorkspace/wallet</value>
    </custom-attribute>
    <custom-attribute name="bipublisher-password-alias">
      <value>bipPwdAlias</value>
    </custom-attribute>
    <custom-attribute name="bipublisher-shared-reports-folders">
      <value>Guest</value>
    </custom-attribute>
  </custom-attributes>
  <parameters>
    <parameter name="_xmode">
      <value>2</value>
    </parameter>
  </parameters>
</bip-reports-work-item>

```

The following is a summary of custom attributes and query string parameters that are configured in the work item element for each reporting tool. ORW uses some of these configuration values to query the reporting tools for a hierarchical list of all the reports to which a user has access. Other configuration values are used to construct the URLs for those reports. The names of the reports are displayed in the navigation panel as hyperlinks to those URLs. When a report's link is clicked, the report launches in the content area of the screen.

Oracle BI EE Work Item Configuration Options

<custom-attribute> options:

Table 8–1

Custom Attribute Name	Description
biee-webservices-url	<p>This is the prefix for the Oracle BI EE Web Services URL e.g. <code>http://<servername>:<portnumber>/analytics/services</code>.</p> <p>This is used by Workspace to call the Oracle BI EE <code>SAWSessionService</code> and <code>WebCatalogService</code> web services in order to query the list of reports the logged-in user has access to. If the Oracle BI EE analytics URL is SSO protected, the services URL must not be protected or should be explicitly unprotected.</p> <p>For more details see the "Integrating with Report Servers" section of Chapter 6, "Integration Methods and Communication Flow".</p>
biee-reports-url-prefix	<p>This prefix (e.g. <code>http://<servername>:<portnumber>/analytics/saw.dll</code>) is used by Workspace to compose the first (common) portion of each report's URL. The composed URL is then provided as a link in the Reports work list of Workspace's navigation panel.</p>
biee-logon-id	<p>This is an ID defined as a Oracle BI EE administrator or impersonator. This is so it is capable of accessing the shared folders and any user folders on behalf of the logged-in user.</p>
biee-password-wallet-location (optional together with the bieee-password-alias)	<p>This is the location on the server (where Workspace is installed) of the wallet containing the encrypted password of the <code>biee-logon-id</code>. This location is specified at Workspace install time.</p> <p>If this parameter and the alias are omitted or empty, it is assumed that the <code>biee-logon-id</code> has no password.</p>
biee-password-alias (optional together with the bieee-password-wallet-location)	<p>This is the <code>biee-logon-id</code> password's alias specified at the ORW install when the wallet is created.</p> <p>If this parameter and the wallet location are omitted or empty, it is assumed that the <code>biee-logon-id</code> has no password.</p>
biee-shared-reports-folders (optional)	<p>Example: Shared, Common, etc... This attribute contains a folder or folders defined in the Oracle BI EE tool under the Shared folders. This attribute may contain multiple shared folders that are comma separated.</p> <p>This attribute may be left out or its value may be left empty. In this case no corresponding folder will show up in the Reports work list of the Workspace navigation panel.</p> <p>Example:</p> <pre><custom-attribute name="biee-shared-reports-folders"> <value>shared</value> </custom-attribute></pre>

Table 8–1 (Cont.)

Custom Attribute Name	Description
bieee-users-folders (optional)	<p>This attribute contains the folder name where the Oracle BI EE tool stores its users' folders. For example if this folder is called "Users" and the logged in user name is executive, this user's folders are found in /Users/executive.</p> <p>This attribute may be left out or its value may be left empty. In this case it is assumed that Oracle BI EE stores its users' folders directly under the user's folder e.g. /executive.</p> <p>Example:</p> <pre><custom-attribute name="bieee-users-folder"> <value>users</value> </custom-attribute></pre>

<parameter> options:

Table 8–2

Parameter Name	Description
options	<p>This optional parameter controls links that show up at the bottom of the Oracle BI EE reports.</p> <p>Example:</p> <pre><parameter name="options"> <value>rfd</value> </parameter></pre> <p>The string is comprised of one or more of the following characters:</p> <p>The following list contains the legal values and their descriptions.</p> <ul style="list-style-type: none"> ▪ r (refresh) ▪ f (print) ▪ d (download)

BI Publisher Work Item Configuration Options

<custom-attribute> options:

Table 8–3

Custom Attribute Name	Description
bipublisher-webservices-url	<p>This is the prefix for the BIP Web Services URL e.g. http://<servername>:<portnumber>/xmlpsrver/services.</p> <p>This is used by Workspace to call the BI Publisher ServiceGateway web service in order to query the list of reports the logged-in user has access to. If the BIP server's URL is SSO protected, the services URL must not be protected or should be explicitly unprotected. For more details, see Chapter 6, "Integration Methods and Communication Flow".</p>
bipublisher-reports-url-prefix	<p>This prefix (e.g. http://<servername>:<portnumber>/xmlpsrver) is used by Workspace to compose the first portion of each report's URL. The composed URL is then provided as a link in the Reports work list of Workspace's navigation panel.</p>

Table 8–3 (Cont.)

Custom Attribute Name	Description
bipublisher-logon-id	This is an ID that needs to be defined under the BI Publisher administrator role. This is so it is capable of accessing the shared folders and any user folders on behalf of the logged-in user.
bipublisher-password-wallet-location (optional together with the bipublisher-password-alias)	This is the location on the server (where ORW is installed) of the wallet containing the encrypted password of the bipublisher-logon-id. This location is specified at the time of ORW installation. If this parameter and the alias are omitted or empty, it is assumed that the bipublisher-logon-id has no password.
bipublisher-password-alias (optional together with the bipublisher-password-wallet-location)	This is the bipublisher-logon-id password's alias specified at the time of ORW installation. If this parameter and the wallet location are omitted or empty, it is assumed that the bipublisher-logon-id has no password.
bipublisher-shared-reports-folders (optional)	Example: Guest, Shared, Common, etc... This attribute contains a folder or folders defined in the BI Publisher tool under the Shared Folders. This attribute could contain multiple shared folders that are comma separated. If access to BI Publisher is setup as Guest access, this attribute should be setup as the name of the folder defined in the BI Publisher Admin tab in the Guest Access section. This attribute could be left out or its value could be left empty. In this case no corresponding folder will show up in the Reports work list of the Workspace navigation panel. Example: <pre><custom-attribute name="bipublisher-shared-reports-folders"> <value>Guest</value> </custom-attribute></pre>

<parameter> options:

Table 8–4

Parameter Name	Description
_xmode	This optional parameter controls the header area of the BI Publisher reports. Example: <pre><parameter name="_xmode"> <value>2</value> </parameter></pre> The following list contains the legal values and their descriptions. 1 - Suppresses the logo area, the report parameters area and the Send button. 2 - Suppresses the logo area keeps the report parameters area and shows the template area, the View, Export and Send buttons. 3 - Same as 2 4 - Suppresses all headers and just shows the report.

Note that other parameters that affect the BIP reports behavior can be added. See the Oracle Business Intelligence Publisher documentation for more details.

Adding New Content to the Navigation Panel

Adding Work Lists to the Navigation Panel

You may add new work lists to the navigation panel. You may use either the `<work-list>` or the `<secure-work-list>` element to define your work lists. If you define a work list using `<secure-work-list>`, the work list will be secure and you must use the Permissions Management tool to assign permission grants to the work list. Only users that are members of roles that have been granted permission will be able to view the work list in the navigation panel.

Work lists are added as children of the `<work-lists>` element of the `<navigation-list>` element:

```
<navigation-list id="thenavlist">
  <work-lists>
    <work-list id="DashboardWorkList"
      display-string="{confMsgs.dashboardsWorklist}">
      <icon-uri>#{confMsgs.dashboardWorkListIcon}</icon-uri>
      <work-items>
        ... child work items go here ...
      </work-items>
    ... more work lists go here ...
  </work-lists>
  <homes>
    ... home work items defined here ...
  </homes>
</navigation-list>
```

Simply add or insert `<work-list>` or `<secure-work-list>` elements before or after existing work list elements.

To define a work list, specify the following work list attributes:

- `id="<unique id string>"`—make sure your ID is unique. The ID string is used when assigning permission grants for secure work items.
- `display-string="<work list title>"`—used to define the work list's title. This may be an EL expression that references a tag in a resource bundle.
- `rendered="true"` (or allow to default)

You may also specify an optional icon for the work list by adding an `<icon-uri>` element inside the work list element. The value of the `<icon-uri>` element should be the path of an icon image that has been copied to the "images" directory of the deployed ORW application. For example, if you have created an image named `MyWorkListIcon.gif` and copied it to the "images" directory, you would specify the icon URI element as:

```
<icon-uri>images/MyWorkListIcon.gif</icon-uri>
```

You may specify the `<icon-uri>` element using an EL expression that references a tag in a resource bundle. In that case the value in the resource bundle should be the path to the icon image.

Refer to [Chapter 2, "Backend System Administration and Configuration"](#) for more information about work list configuration parameters.

Changing Work List Folders

You may re-arrange the folders inside a work list. You may add new folders or consolidate multiple folders into a single folder.

You may use either the `<work-item>` or `<secure-work-item>` element to define a folder. If you define a folder using `<secure-work-item>`, you must use the Permissions Management tool to define permission grants for the folder work item. Work items act as folders if they have "child" work items defined. Typically folder work items do not have launchable content associated with them.

To define a folder work item, specify the following work item attributes:

- `id="<unique id string>"`—make sure your ID is unique. The ID string is used when assigning permission grants for secure work items.
- `display-string="<folder label>"`—used to define the folder's label. This may be an EL expression that references a tag in a resource bundle.
- `rendered="true"` (or allow to default)
- `launchable="false"` (or allow to default)—typically folder work items do not have launchable content.

Specify the following work item child elements:

- `child-work-items`—the work items that are contained in the folder are defined as children of the `<child-work-items>` element. The child work items may be either `<work-item>` or `<secure-work-item>` elements.

Adding Work Items to the Navigation Panel

You may add new work items to the navigation panel. You may add work items to work lists or to "folder" work items. You may use either the `<work-item>` or the `<secure-work-item>` element to define your work items. If you define a work item using `<secure-work-item>`, the work item will be secure and you must use the Permissions Management tool to assign permission grants to the work item. Only users that are members of roles that have been granted permission will be able to view the work item in the navigation panel.

Work items are added as children of the `<work-items>` element of the `<work-list>` or `<secure-work-list>` element:

```
<work-list id="ApplicationsWorklist"
  display-string="#{confMsgs.applicationsWorklist}">
  <icon-uri>#{confMsgs.appsWorkListIcon}</icon-uri>
  <work-items>
    <secure-work-item id="SSOApplicationsFolder"
      display-string="#{confMsgs.ssoAppFolder}"
      rendered="true">
      ... work item child elements go here ...
    </secure-work-item>
    ... more work items go here ...
  </work-items>
</work-list>
```

Work items may also be added as children of the `<child-work-items>` element of a "folder" `<work-item>` or `<secure-work-item>`:

```

<secure-work-item id="NonSSOStoresFolder"
    display-string="#{confMsgs.storesAppFolder}"
    rendered="true">
  <child-work-items>
    <secure-work-item id="centralOffice"
        display-string="#{confMsgs.centralOfficeTitle}"
        rendered="true"
        launchable="true"
        show-in-content-area="false">
      <url>URL to Central Office application here</url>
      <parameters>
        <parameter name="EXAMPLE_PARAMETER">
          <value>EXAMPLE_VALUE</value>
        </parameter>
      </parameters>
    </secure-work-item>
  </child-work-items>
</secure-work-item>

```

To define a work item, specify the following work item attributes:

- `id="<unique id string>"` —make sure your ID is unique. The ID string is used when assigning permission grants for secure work items.
- `display-string="<work item label>"`—used to define the work item's label. This may be an EL expression that references a tag in a resource bundle.
- `rendered="true"` (or allow to default)

The values of the following attributes depend on the type of work item you are creating:

- `launchable`—Set this attribute to "true" if the work item should be a link that launches content, either in the content area or in a new browser window.
- `show-in-content-area`—Set this attribute to "true" if the work item is launchable and the content should be displayed in the ORW content area. Set this attribute to false if the work item is launchable and the content should be launched in a new browser window.
- `target-frame`—Set this attribute if your launchable content requires you to specify a target frame. For example, the Oracle Price Management (RPM) application uses Java WebStart to launch the application. RPM requires `target-frame` to be set as `"_self"` to keep a blank browser window from appearing before WebStart launches the application.

Note: Dashboards developed for the ORW application must have the `target-frame` attribute set to one of the following values:

- `target-frame="_iframe"`—This option specifies that the dashboard will be displayed in the content area in an HTML `<iframe>` tag.
- `target-frame="_launchable_iframe"`—This option is similar to `"_frame"`. It tells ORW to display the content within the content area in an HTML `<iframe>` tag. In addition, an "Open in New Window" hyperlink is displayed above the content. This hyperlink will launch the content in a new window.

Depending on the type of work item you are creating, you may also need to configure additional configuration parameters in the form of child XML elements:

- `<url>`—This element is required if you have specified `launchable="true"`. Use this element to specify the URL of the content to launch.

- `<parameters>`—This element is used to specify a list of query string parameters for the URL. Each parameter is configured in a `<parameter>` element:

```
<parameter name="TOP_REPORT_TITLE">
  <value>#{storesDashboardMsgs.topReportTitle}</value>
</parameter>
```

Refer to ["Defining URL Query String Parameters for a Work Item"](#) for more information about configuring query string parameters.

- `<custom-attributes>`—This element is used to specify a list of custom attributes for the work item. Specialized types of work items can define custom attributes that may be defined for that work item.
 - Example 1: Secure work items use the `adf-permission-target` to configure the name of an ADF "page def" file that requires an ADF Region Permission to be configured in the Permissions Management tool.
 - Example 2: Oracle BI EE and BIP reports work items have several custom attributes that are used to customize the entries in the Reports work list. Refer to [Customizing Reports Work Items](#) for more details.

Refer to [Chapter 2, "Backend System Administration and Configuration"](#) for more information about work item configuration parameters. Refer to the *Oracle Retail Workspace Administration Guide* for more information about the Permissions Management tool.

Defining URL Query String Parameters for a Work Item

When you configure a work item to launch content, you specify the URL to launch with the `<url>` element. Some URLs require additional query string parameters. For example:

```
http://mycompany.com:7780/MyPage?config=MyConfig&template=MyTemplate
```

You may configure the list of query string parameters that will be appended to the URL by specifying a `<parameters>` element. Each parameter is defined in a `<parameter>` element:

```
<parameters>
  <parameter name="config">
    <value>MyConfig</value>
  </parameter>
  <parameter name="template">
    <value>MyTemplate</value>
  </parameter>
</parameters>
```

ORW assembles the query string in the order that the parameters are configured, and then appends the query string to the URL. ORW performs URL encoding for the parameter values before appending the parameter value to the query string. It does not perform URL encoding for the parameter "name" attribute. ORW supports only ASCII characters for parameter names.

Adding Work Items that Launch a URL in the Content Area

You may configure work items to launch content in the ORW content area. ORW is able to display web pages in the content area if they display correctly inside of an HTML `<iframe>` tag.

After adding the work item as specified in [Adding Work Items to the Navigation Panel](#), configure the URL to launch using the `<url>` element. If necessary, configure URL query string parameters using the `<parameters>` element. Then make sure you have specified the following attribute values in the `<work-item>` or `<secure-work-item>` element:

- `launchable="true"`
- `show-in-content-area="true"`
- `target-frame="_iframe"` or `"_launchable_iframe"`

For secure work items, after saving your configuration changes you must log in to ORW as an administrator that has permissions to run the Permissions Management tool. Use the Permissions Management screen to assign permission grants for the work item.

Adding Work Items that Launch a URL in a Browser Window

You may configure work items to launch content in a separate browser window. After adding the work item as specified in [Adding Work Items to the Navigation Panel](#), configure the URL to launch using the `<url>` element. If necessary, configure URL query string parameters using the `<parameters>` element. Then make sure you have specified the following attribute values in the `<work-item>` or `<secure-work-item>` element:

- `launchable="true"`
- `show-in-content-area="false"`
- `target-frame`—Configure this value only if required by the web page. This attribute specifies the "target" of the HTML Anchor `<A>` element that is used to render the link in the navigation panel. If no `target-frame` attribute is specified, the default is the `"_blank"` target, which results in launching the URL in a new unnamed window.

For secure work items, after saving your configuration changes you must log in to ORW as an administrator that has permissions to run the Permissions Management tool. Use the Permissions Management screen to assign permission grants for the work item.

Adding a Secure Dashboard to the Dashboards List

The basic steps for adding a secure dashboard are similar to the steps for adding work items that launch content in the content area (see ["Adding Work Items that Launch a URL in the Content Area"](#)). In addition you must also define a `<custom-attribute>` element that defines the `adf-permission-target`. The `adf-permission-target` custom attribute must specify the fully-qualified name of the ADF "page def" file associated with your secure dashboard:

```
<secure-work-item id="myDashboardUniqueId"
    display-string="#{customMsgs.myDashboardLabel}"
    rendered="true"
    launchable="true"
    show-in-content-area="true"
    target-frame="_iframe">
  <url>URL to deployed dashboard</url>
  <parameters>
    ... query string parameter elements go here ...
  </parameters>
  <custom-attributes>
```

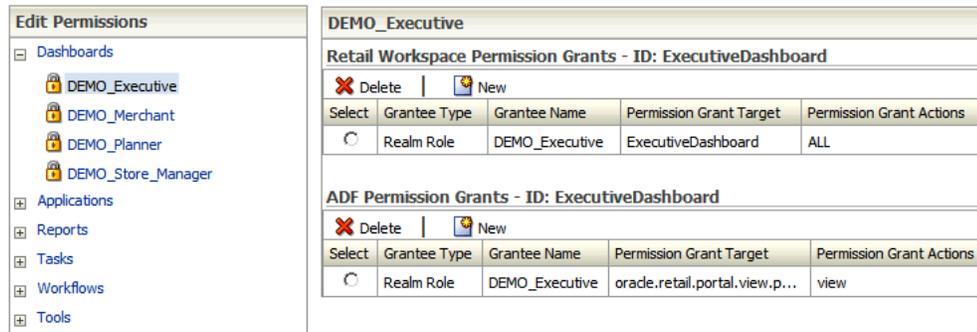
```

<custom-attribute name="adf-permission-target">
  <value>mydashboard.view.pageDefs.MyDashboardPageDef</value>
</custom-attribute>
</custom-attributes>
</secure-work-item>

```

After saving your configuration changes you must log in to ORW as an administrator that has permission to access the Permissions Management tool. Use the Permissions Management screen to assign permission grants for the work item. You must add permission grants in both the ORW and ADF lists:

Figure 8–9 Permissions Management Screen



Refer to [Chapter 9, "Dashboard Development Tutorial"](#) for more details regarding creating and configuring a custom dashboard.

Removing Content from the Navigation Panel

Removing Work Items

If a work item or any of its descendent work items are secure, you first should use the Permissions Management tool to delete ORW permission grants for the secure work item(s) you plan to delete. Once there are no ORW permission grants defined for the work item, you may remove the work item by deleting the work item element and all of its child elements from `retail-workspace-page-config.xml`.

If you remove a work item that is also configured as a home work item, you must also remove the reference to the work item from the homes list. If you fail to do so, the ORW application will be unable to parse the XML configuration file and will report an exception when a user attempts to access the application's URL.

When you remove work items that you have customized, keep in mind that you may want to also clean up any other resources you created for that work item. For example, if the work item uses internationalized strings that you defined in a custom resource bundle, you may want to remove the strings from the resource bundle. If the custom resource bundle will no longer be in use, you also may want to delete the bundle and remove any resource-bundle elements that reference the bundle from `retail-workspace-page-config.xml`.

Example: Removing an unsecure work item

In this example, you have decided to remove the Metalink work item from the Tools work list. In the default configuration, the Metalink work item is unsecure.

1. Locate the Metalink work item in `retail-workspace-page-config.xml`. The work item will look similar to the following fragment of XML:

```
<work-item id="userMetalink"
  display-string="#{confMsgs.metalinkTitle}"
  rendered="true"
  launchable="true"
  show-in-content-area="false" >
  <url>http://metalink.oracle.com</url>
</work-item>
```

2. Remove the work item from the configuration. Delete the entire work-item element, from beginning "`<work-item`" through ending "`</work-item>`".

Example: Removing an Oracle Retail application from the Applications work list

In this example, you have decided to remove the Merchandising System (RMS) application from the Applications worklist. Since RMS is a secure work item, before you can delete it from the configuration, you must remove permission grants defined for RMS.

1. In the ORW application, log in as a user that has permission to execute the Permissions Management tool. Navigate to the Permissions Management tool in the navigation panel and select it.
2. In the Edit Permissions panel, locate the "Merchandising System (RMS)" node. In the default configuration, the RMS node can be found by expanding Applications/Merchandising. Select the RMS node.
3. Remove the RMS ORW permission grants one by one. Refer to the *Oracle Retail Workspace Administration Guide* for more information regarding the Permissions Management tool.
4. Locate the RMS work item in `retail-workspace-page-config.xml`. The work item will look similar to the following fragment of XML:

```
<secure-work-item id="rms"
  display-string="#{confMsgs.rmsTitle}"
  rendered="true"
  launchable="true"
  show-in-content-area="false">
  <url>http://mycompany.com:7780/forms/frmservlet</url>
  <parameters>
    <parameter name="config">
      <value>rms13devhpss</value>
    </parameter>
  </parameters>
</secure-work-item>
```

5. Remove the work item from the configuration. Delete the entire secure-work-item element, from beginning "`<secure-work-item`" through ending "`</secure-work-item>`".

Example: Removing a Dashboard

In this example, you have decided to remove the Demo Executive Dashboard from the Dashboards worklist. Since the Executive Dashboard is a secure work item, before you delete it from the configuration you should remove permission grants defined for the Executive Dashboard.

In the default configuration, the Executive Dashboard is defined as a home page for the demo_executive role. When you remove the work item from retail-workspace-page-config.xml, you also must remove the <home> element that refers to the Executive Dashboard.

1. In the ORW application, log in as a user that has permission to execute the Permissions Management tool. Navigate to the Permissions Management tool in the navigation panel and select it.
2. In the Edit Permissions panel, locate the executive dashboard node. In the default configuration, the executive dashboard node can be found by expanding the Dashboards folder. Select the executive dashboard node.
3. Remove the executive dashboard permission grants one by one. Refer to the *Oracle Retail Workspace Administration Guide* for more information regarding the Permissions Management tool.
4. Locate the executive dashboard work item in retail-workspace-page-config.xml. The work item will look similar to the following fragment of XML:

```
<secure-work-item id="ExecutiveDashboard"
    display-string="{examplesMsgs.executiveDashboard}"
    rendered="true" launchable="true"
    show-in-content-area="true">
  <url>exampleDashboards/DemoExecutiveDashboard.jsp</url>
  <parameters>
    (... several lines of XML intentionally omitted)
  </parameters>
</secure-work-item>
```

5. Remove the work item from the configuration. Delete the entire secure-work-item element, from beginning "<secure-work-item>" through ending "</secure-work-item>".
6. Locate the list of <home> work items in the retail-workspace-page-config.xml:

```
<homes>
  <!--
    Each 'home' refers to the ID of the work item that
    is the default 'home page' for the users in the specified
    role. If a user is in multiple roles, the first matching
    role found is used. Role names must match the names of
    configured security roles.
  -->
  <home role="demo_executive" value="ExecutiveDashboard"/>
  <home role="demo_merchant" value="MerchantDashboard"/>
  <home role="demo_planner" value="PlannerDashboard"/>
  <home role="demo_store_manager" value="StoresDashboard"/>
</homes>
```

7. Remove the home definition for the executive dashboard. Edit retail-workspace-page-config.xml to delete the home element whose value is a reference to the ID attribute of the dashboard. In this case the <home> element whose value is equal to "ExecutiveDashboard":

```
<homes>
  <!--
    Each 'home' refers to the ID of the work item that
    is the default 'home page' for the users in the specified
    role. If a user is in multiple roles, the first matching
```

```

        role found is used. Role names must match the names of
        configured security roles.
-->
<home role="demo_merchant" value="MerchantDashboard"/>
<home role="demo_planner" value="PlannerDashboard"/>
<home role="demo_store_manager" value="StoresDashboard"/>
</homes>

```

Removing Work Lists

If a work list or any of its descendent work items are secure, you first must use the Permissions Management tool to delete ORW permission grants for the secure work item(s) and secure work list you plan to delete. Once there are no ORW permission grants defined for the work list and any of its descendent work items, you may remove the work list by deleting the work list element and all of its child elements from `retail-workspace-page-config.xml`.

When you remove work lists that you have customized, keep in mind that you may want to also clean up any other resources you created for that work list. For example, if the work list uses internationalized strings that you defined in a custom resource bundle, you may want to remove the strings from the resource bundle. If the custom resource bundle will no longer be in use, you also may want to delete the bundle and remove any resource-bundle elements that reference the bundle from `retail-workspace-page-config.xml`.

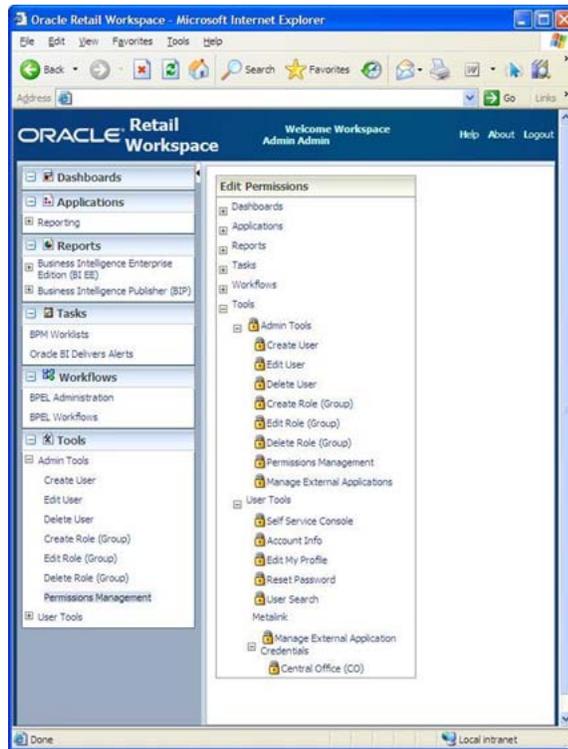
Example: Removing an Unsecure Work List that has Secure Work Item Descendents

In this example, you have decided to remove the Tools work list. In the default configuration, the Tools work list is unsecure, and it has a mix of secure and unsecure descendent work items (the unsecure "User Tools" folders, the secure "Admin Tools" folder, and the various secure work items that are contained in those folders).

1. In the ORW application, log in as a user that has permission to execute the Permissions Management tool. Navigate to the Permissions Management tool in the navigation panel and select it.
2. In the Edit Permissions panel, locate the "Tools" work list. Expand the work list then locate the secure work items below this node. A work item is secure if it has a "lock" icon.
3. For each secure work item that is a descendent of the "Tools" node, select the work item, then use the Permission Management screens to remove the permission grants one by one. Be careful to remove permission grants only for the secure work items that descend from the "Tools" work list. In other words, do not remove permission grants for other work lists.

In the following figure, you will notice the Admin Tools and User Tools folders are expanded, and in each folder are several more secure work items. In this example, you would select each of these work items one by one and use the Permission Management screens to remove the permission grants. Refer to the *Oracle Retail Workspace Administration Guide* for more information regarding the Permissions Management tool.

Figure 8–10 Secured Work Items Under the Admin Tools and User Tools Worklists



4. Locate the Tools work list in `retail-workspace-page-config.xml`. The work list will look similar to the following fragment of XML:

```

<work-list id="ToolsWorklist"
    display-string="#{confMsgs.toolsWorklist}">
  <icon-uri>#{confMsgs.toolsWorkListIcon}</icon-uri>
  <work-items>
    <secure-work-item id="AdminToolsFolder"
      display-string="#{confMsgs.adminToolsFolder}"
      rendered="true">
      <child-work-items>
        <secure-work-item id="adminCreateUser"

          (... several lines of XML intentionally omitted)

        </child-work-items>
      </secure-work-item>
    <work-item id="UserToolsFolder"
      display-string="#{confMsgs.userToolsFolder}"
      rendered="true">
      <child-work-items>
        <secure-work-item id="userSelfService"

          (... several lines of XML intentionally omitted)

        <work-item id="userMetalink"
          display-string="#{confMsgs.metalinkTitle}"
          rendered="true"
          launchable="true"
          show-in-content-area="false" >
          <url>http://metalink.oracle.com</url>

```

```

        </work-item>
    </child-work-items>
</work-item>
</work-items>
</work-list>

```

5. Remove the work list from the configuration. Delete the entire work-list element, from beginning "<work-list id="ToolsWorklist" through ending "</work-list>".

Configuring the Workspace Home Page

You may define a list of "home" work items in `retail-workspace-page-config.xml`. A home work item is the default work item that will be displayed in the content area of ORW.

Each home work item's value is a reference to the ID of a work item that is configured elsewhere in the configuration file. Each home work item is associated with a security role (OID group). After installing ORW, the home page configuration looks similar to this:

```

<homes>
  <!--
    Each 'home' refers to the ID of the work item that
    is the default 'home page' for the users in the specified
    role. If a user is in multiple roles, the first matching
    role found is used. Role names must match the names of
    configured security roles.
  -->
  <home role="demo_executive" value="ExecutiveDashboard"/>
  <home role="demo_merchant" value="MerchantDashboard"/>
  <home role="demo_planner" value="PlannerDashboard"/>
  <home role="demo_store_manager" value="StoresDashboard"/>
</homes>

```

When the ORW application is opened, or after a user logs in or out, the application evaluates the "homes" list to determine which home work item to display in the content area.

First ORW gets a list of all security roles that the current user belongs to. Note that all users, including an un-authenticated user (one that hasn't logged in) also belong to a logical role called "anyone". For each role, ORW checks to see if there is a work item in the "homes" list whose "role" attribute matches (note: role name comparisons are case insensitive). If a match is found, ORW then checks to make sure the user is in a role that has permission to view the work item's content. ORW also checks if the work item's "rendered" attribute is equal to "true". The first home work item that meets all of these criteria is displayed in the ORW content area. If a matching home work item is not found, the content area is empty.

Adding a Home Work Item

To add a new home work item, insert a <home> element inside the <homes> element. The role attribute should match the name of a security role defined in OID. The value attribute must match exactly the ID of a work item that is defined elsewhere in `retail-workspace-page-config.xml`. If the value does not match the ID of an existing work item, the ORW application is unable to parse the XML configuration file and reports an exception when a user attempts to access the application. Make sure the specified role has been granted permission to access the work item. The Permission Management tool may be used to view and change work item permissions.

Deleting a Home Work Item

To delete a home work item, simply remove the home XML element. Deleting the home element does not remove the actual work item that it references.

Editing a Home Work Item

You may edit an existing home work item to change either the role and/or value attributes.

When changing the role attribute, the new value should match the name of a security role (group) defined in OID. The role attribute value is case insensitive.

When changing the value attribute, make sure to change it to the ID of a work item that is defined elsewhere in `retail-workspace-page-config.xml`.

When changing either role or value, make sure the specified role has been granted permission to access the work item.

Developing and Using Custom Configuration Elements

The ORW configuration file provides a great deal of flexibility for customizing the navigation panel. In certain situations the out-of-the-box capabilities of `retail-workspace-page-config.xml` may not completely suit your needs. For those cases, ORW provides facilities for developing custom ORW configuration elements.

For example, you may have a third party reporting tool that can be queried to get a hierarchical list of reports. You may wish to develop an extended version of the `<work-item>` element that dynamically retrieves the report list as a sub-tree of work items.

The following skills and resources are necessary to develop custom configuration elements for ORW:

- Thorough knowledge of `retail-workspace-page-config.xml`.
- A Java programming toolkit, such as Oracle JDeveloper.
- Thorough knowledge of Java programming.
- Thorough knowledge of Java ServerFaces, ADF Faces and the JSF lifecycle.
- Thorough knowledge of Castor XML data binding, including Castor mapping files.

ORW Configuration Framework Design Overview

A prerequisite for developing custom configuration elements is an understanding of the ORW configuration framework. The ORW configuration framework makes use of Castor XML, an XML data binding framework that is a subset of the Castor open source data binding framework for Java. ORW uses the Castor XML data binding framework to convert an XML configuration file to Java Objects. For detailed information about Castor XML data binding, refer to <http://www.castor.org>.

In the case of `retail-workspace-page-config.xml`, the ORW configuration framework uses Castor XML to unmarshal the configuration file data stream into a hierarchy of Java Objects, with an instance of the `oracle.retail.portal.framework.ui.model.impl.RetailPortalPage` class as the root of the hierarchy.

Castor XML provides an implied set of rules for unmarshalling, called introspection mode, that uses Java reflection to determine the binding between XML and Java classes. Castor XML also provides a mapping mode, in which the developer specifies a mapping file that defines the mapping between XML and Java classes. The ORW configuration framework makes use of mapping mode to provide a configurable set of rules to control unmarshalling of ORW configurations. The ORW application includes a default standard mapping configuration. ORW also allows specification of a custom mapping configuration, in which individuals who customize ORW may override the mapping configuration to include mapping for custom classes.

Steps for Developing and Using Custom Configuration Elements - Overview

Development of custom configuration elements is a multi-step process:

- Determine the ORW configuration element to extend.
Typically you will extend the `<work-item>` or `<secure-work-item>` element.
- Develop a Java class that extends the ORW class that implements the configuration element you are extending.
- Define a Castor mapping file that describes the XML that will be used to define the elements and attributes used to configure your extended configuration element.
- Deploy your custom classes, resource bundles and mapping file to the ORW OC4J instance.
- Customize ORW's mapping configuration to reference your custom class' Castor mapping file.
- Change `retail-workspace-page-config.xml` to use your custom configuration element.

Developing Custom Work Items

In most cases the contents of an ORW worklist may be defined by configuring a hierarchy of `<work-item>` or `<secure-work-item>` elements in `retail-workspace-page-config.xml`. For the rare cases where `<work-item>` or `<secure-work-item>` do not provide the desired behavior, you may develop a custom work item. Custom work items may be implemented as extensions of the following ORW classes:

- `oracle.retail.portal.framework.ui.model.impl.WorkItem`
- `oracle.retail.portal.framework.ui.model.impl.SecureWorkItem`

Custom work items are developed by extending public methods of the `WorkItem` or `SecureWorkItem` class. Custom work items may make use of the `<custom-attributes>` element of `WorkItem` to pass developer-defined variable attributes from the ORW configuration to the work item instance. The custom work item class retrieves custom attributes by calling the `getCustomAttributes()` method.

Example: Developing a custom work item

In the following example, we have a requirement for an ORW worklist whose contents will change frequently. We would like to discover the changes to the work list dynamically rather than making frequent changes to `retail-workspace-page-config.xml`, so we have developed a service for cataloging our content. We have decided to create a new class that extends `SecureWorkItem`, `com.mycompany.MyCustomWorkItem`. `MyCustomWorkItem` will dynamically get its hierarchy of child work items from our new service. We also will

create two new classes to represent the child work items:
`com.mycompany.FileWorkItem`, which extends `WorkItem`, and
`com.mycompany.DirectoryWorkItem`, which extends `FileWorkItem`.

1. Set up your development environment to include necessary jar files.

Since custom work item classes extend from either `WorkItem` or `SecureWorkItem`, you must include two ORW code jars in the development environment. Extract the following jars from the `orw-devtools.zip` file and include them as libraries when compiling your custom work item classes:

- `oretail-pageFramework.jar`
- `oretail-common.jar`

2. Create a resource bundle that includes all internationalized strings. For this example, create a file named `com.mycompany.MyMessages` with the string that will be returned from `MyCustomWorkItem.getDisplayString()`:

```
myCustomWorkItemDisplayString=Custom Test Work Item
```

3. Code and compile your custom work item classes.

`MyCustomWorkItem` will be the root node of our work list. We have decided to override the following `SecureWorkItem` methods in `MyCustomWorkItem`:

- `public List getChildren()`

Our version of `getChildren()` will dynamically get the top level list of children from our service, instantiate the appropriate child work item for each item in the top level of the hierarchy (`DirectoryWorkItem` or `FileWorkItem`), and add each work item to `MyCustomWorkItem`'s list of children.

- `public boolean isLaunchable()`

There is no content associated with `MyCustomWorkItem`, so we have decided to override `isLaunchable()` to always return false.

- `public String getDisplayString()`

We do not want to allow configuration of the display string for `MyCustomWorkItem`, so we will override `getDisplayString()` to return our preferred display string. We have decided to internationalize the display string, so we will define a resource bundle that defines the string.

`FileWorkItem` will be used to represent "leaf" nodes in `MyCustomWorkItem`'s hierarchical list of child nodes. `FileWorkItems` are work items that do not have children. We have decided to override the following `WorkItem` methods in `FileWorkItem`:

- `public String getDisplayString()`

Our class overrides `getDisplayString()` to return the name of the file node. We will get the name of the file node from our service.

- `public String getURL()`

We want to launch content associated with the file work item, so we are overriding `getURL()` to return the URL of the content. The URL must be content that can be displayed in the ORW content area's iframe. We will get the URL for the work item from our service.

- `public boolean isLaunchable()`

We want to launch content associated with the file work item, so we are overriding `isLaunchable()` to return true.

- `public boolean isShowInContentArea()`

We want to launch the file work item's content in the ORW content area, so we are overriding `isLaunchable()` to return true.

- `public String getTargetFrame()`

We want to launch the file work item's content within an iframe in the ORW content area, so we are overriding `getTargetFrame()` to return `"_iframe"`.

`DirectoryWorkItem` will be used to represent folders in `MyCustomWorkItem`'s hierarchical list of child nodes. `DirectoryWorkItems` are work items that can have children. Since `DirectoryWorkItem` extends `FileWorkItem`, it inherits its methods, as well as those from `WorkItem`. We have decided to override the following methods in `DirectoryWorkItem`:

- `public List getChildren()`

Each `DirectoryWorkItem` is responsible for querying our service for its list of children. Our version of `getChildren()` will query the service to dynamically get the `DirectoryWorkItem`'s list of children. It will add an appropriate child work item for each directory or file.

- `public int getChildCount()`

Since we are overriding `getChildren()`, we also must override `getChildCount()`. In our version of the method we will call `getChildren()` to make sure children have been loaded before calling `super.getChildCount()` to get the child count.

- `public WorkElement getChild(int)`

Since we are overriding `getChildren()`, we also must override `getChild(int)`. In our version of the method we will call `getChildren()` to make sure children have been loaded before calling `super.getChild(int)` to get the child at the specified index.

- `public String getURL()`

Since our directory work items do not have associated launchable content, we are overriding `getURL()` to return false.

- `public boolean isLaunchable()`

Since our directory work items do not have associated launchable content, we are overriding `isLaunchable()` to return false.

- `public boolean isShowInContentArea()`

Since our directory work items do not have associated launchable content, we are overriding `isShowInContentArea()` to return false.

Following are snippets from our example custom `FileWorkItem` class:

```
package com.mycompany;

import oracle.retail.portal.framework.ui.model.WorkElement;
import oracle.retail.portal.framework.ui.model.impl.WorkItem;

public class FileWorkItem extends WorkItem {

    // Constructor omitted. The constructor should perform
    // necessary initialization
```

```

public String getDisplayString() {
    // This method will return the name of the file.
    // Code omitted.
}

public String getURL() {
    // This method will return the URL of the file.
    // Code omitted.
}

public boolean isLaunchable() {
    // The file work item is launchable, so this method
    // always returns true
    return true;
}

public boolean isShowInContentArea() {
    // The file work item is launchable, so this method
    // always returns true
    return true;
}

public String getTargetFrame() {
    // The file work item is launchable in the content
    // area iframe, so this method always returns "_iframe"
    return "_iframe";
}
}

```

Following are snippets from our example custom `DirectoryWorkItem`:

```

package com.mycompany;

import java.util.List;

import oracle.retail.portal.framework.ui.model.WorkElement;
import oracle.retail.portal.framework.ui.model.impl.WorkItem;
public class DirectoryWorkItem extends FileWorkItem {
    // Constructor omitted. The constructor should perform
    // necessary initialization

    public boolean isLaunchable() {
        // The directory work item is not launchable, so this method
        // always returns false
        return false;
    }

    public String getURL() {
        // The directory work item is not launchable, so this method
        // always returns a null URL
        return null;
    }

    public boolean isShowInContentArea() {
        // The directory work item is not launchable, so this method
        // always returns false
        return false;
    }

    public int getChildCount() {

```

```

        // make sure children are loaded
        getChildren();
        return super.getChildCount();
    }

    public WorkElement getChild(int i) {
        // make sure children are loaded
        getChildren();
        return super.getChild(i);
    }

    public List getChildren() {
        // When overriding getChildren(), call the superclass
        // getChildren() to allow the superclass to instantiate the
        // List if necessary.
        List children = super.getChildren();
        // This example is clearing the child list every time,
        // then rebuilding it. It is also possible to load the
        // children the first time this method is called, then
        // return cached data in subsequent calls.
        children.clear();

        // At this point we would query the service for the
        // information about the child directories and files.
        // Following is some pseudocode that shows how it might
        // be done:

        // Call service to get the top level children. Lets assume that
        // the service returns a list of objects that encapsulate
        // information about each top level child.
        List children = // code for calling the service goes here
        int childCount = children.size();

        // Now iterate through the children. Following is a simple 'for'
        // loop but other constructs could be used.
        for (int i = 0; i < childCount; i++) {
            // If the child object is a 'directory', our example creates
            // a DirectoryWorkItem
            if (test for directory goes here) {
                children.add(new DirectoryWorkItem(<parameters go here>);
            } else {
                children.add(new FileWorkItem(<parameters go here>);
            }
        }
        return children;
    }
}

```

Following are snippets from our example custom `MyCustomWorkItem` class:

```

package com.mycompany;

import java.util.List;

import java.util.ResourceBundle;

import javax.faces.context.FacesContext;

import oracle.retail.portal.framework.ui.model.impl.SecureWorkItem;

public class MyCustomWorkItem extends SecureWorkItem {

```

```

public MyCustomWorkItem() {
}

public List getChildren() {
    List children = super.getChildren();
    children.clear();

    // Our example passes a "root_file" variable to the
    // custom work item. This is accomplished by configuring a
    // <custom-attributes> element with a child element that defines
    // the "root_file" custom attribute. The following sample code
    // shows how to retrieve the value of the "root_file" custom attribute
    String rootFilePath = getCustomAttributes().get("root_file");

    // At this point the method would call the service to
    // get the information about the root node. Then it
    // would instantiate an appropriate root work item, in this
    // case a DirectoryWorkItem, and add it to the list of
    // children...

    children.add(new DirectoryWorkItem(<parameters go here>));

    return children;
}

public boolean isLaunchable() {
    // This work item is not launchable, so this method
    // always returns false
    return false;
}

public String getDisplayString() {
    // In this example, we are getting the display string
    // from our resource bundle. We are getting the Locale
    // from the Faces view root.
    FacesContext context = FacesContext.getCurrentInstance();
    ResourceBundle bundle =
        ResourceBundle.getBundle("com.mycompany.MyMessages",
            context.getViewRoot().getLocale());
    return bundle.getString("myCustomWorkItemDisplayString");
}
}

```

4. Develop a Castor mapping file for any custom work item that will be configured in `retail-workspace-page-config.xml`. The Castor mapping file describes the XML syntax that will be used to configure the custom work item. In our example, we will provide a mapping file only for the `MyCustomWorkItem` class. The other classes do not require mapping because they will not be configured in `retail-workspace-page-config.xml`.

For our example, we will create `MyCustomWorkItem-mapping.xml` in the same directory as the class, `com/mycompany/`.

Since `MyCustomWorkItem` extends `SecureWorkItem`, its mapping file will extend the mapping for `SecureWorkItem` that is included in the ORW standard mappings.

The following XML shows the contents of `MyCustomWorkItem-mapping.xml`:

```

<?xml version="1.0" encoding="UTF-8"?>
<mapping xmlns="http://castor.exolab.org/"
        xmlns:cst="http://castor.exolab.org/">

```

```

<description>example mapping file for class that extends
    SecureWorkItem</description>
<class name="com.mycompany.MyCustomWorkItem"
    identity="id"
    extends="oracle.retail.portal.framework.ui.model.impl.SecureWorkItem">
    <description>default mapping for MyCustomWorkItem</description>
    <map-to xml="test-custom-work-item"/>
</class>
</mapping>

```

In the example above, the `<map-to xml="test-custom-work-item"/>` defines "test-custom-work-item" as the name of the element to use to configure the custom work item.

For more information about Castor XML mapping files, refer to <http://www.castor.org>.

5. Deploy the custom work item classes, resource bundle(s) and mapping file(s).

Custom class files and mapping files are deployed by copying them to the `custom_classes` directory of the ORW application (`<ORACLE_HOME>/j2ee/<oc4j_instance>/RetailWorkspace/custom_classes`), The Java package directory structure of the classes and mapping files must be maintained. For our example all of our classes and resources are in the `com.mycompany` Java package. We will copy the following files to `custom_classes/com/mycompany`:

- `MyCustomWorkItem.class`
- `DirectoryWorkItem.class`
- `FileWorkItem.class`
- `MyCustomWorkItem-mapping.xml`

Any resource bundles must also be deployed. To be consistent with other ORW resource bundles, copy any new resource bundles to the `lang_packs` directory of the ORW application (`<ORACLE_HOME>/j2ee/<oc4j_instance>/RetailWorkspace/lang_packs`). For our example, copy `MyMessages.properties` to `lang_packs/com/mycompany`. Alternatively you may deploy the resource bundle to a jar file and copy the jar to the ORW `lang_packs` directory.

6. Create a custom mapping configuration.

Before you can use your new custom work item, you must tell ORW where to find the mapping files for your custom classes. ORW includes standard mapping configurations that define the default mapping files used by the ORW configuration classes. To customize the ORW `RetailPortalPage` mapping configuration, extract the following ORW `RetailPortalPage` mapping configuration from the `mapping-config-templates` subdirectory of `orw-devtools.zip` to a temporary location:

```

oracle.retail.portal.framework.ui.model.impl.RetailPortalPage-mapping-config.xml
1

```

The default version of the `RetailPortalPage` mapping configuration that you have just extracted will look similar to the following XML (file indentation may vary):

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
    Copyright (c) 2008, 2009, Oracle and/or its affiliates. All rights
    reserved.

```

```

    $Id:

oracle.retail.portal.framework.ui.model.impl.RetailPortalPage-mapping-config.xml
l,v 1.1 2009/04/08 21:47:13 christv Exp $
-->
<mapping-list base-url="oracle/retail/portal/framework/ui/model/impl/config/
RetailPortalPageConfig-mapping.xml">
  <mapping-file>
    oracle/retail/portal/framework/ui/model/impl/config/RetailPortalPageConfig
-mapping.xml
  </mapping-file>
</mapping-list>

```

To add your custom work item, edit `oracle.retail.portal.framework.ui.model.impl.RetailPortalPage-mapping-config.xml` and add another `<mapping-file>` element as a child of the `<mapping-list>` element:

```

<mapping-list base-url="oracle/retail/portal/framework/ui/model/impl/config/
RetailPortalPageConfig-mapping.xml">
  <mapping-file>
    oracle/retail/portal/framework/ui/model/impl/config/RetailPortalPageConfig
-mapping.xml
  </mapping-file>
  <mapping-file>com/mycompany/MyCustomWorkItem-mapping.xml</mapping-file>
</mapping-list>

```

The value of the `<mapping-file>` element should refer to the fully-qualified name of your class's mapping file, in this example `com/mycompany/MyCustomWorkItem-mapping.xml`.

7. Stop the ORW application before deploying the custom mapping configuration and changing the ORW configuration.
8. Deploy the custom mapping configuration.

Copy the customized version of `oracle.retail.portal.framework.ui.model.impl.RetailPortalPage-mapping-config.xml` to the ORW application's `custom_mapping_configs` directory (`<ORACLE_HOME>/j2ee/<oc4j_instance>/RetailWorkspace/custom_mapping_configs`).

9. Change `retail-workspace-page-config.xml`.

Edit `retail-workspace-page-config.xml` to include the new custom work item. In our example case, the work item is specified using a `<test-custom-work-item>` element:

```

<test-custom-work-item id="testCustomWorkItem">
  <custom-attributes>
    <custom-attribute name="root_file">
      <value>/root/file/directory/path</value>
    </custom-attribute>
  </custom-attributes>
</test-custom-work-item>

```

Note that the example specifies the "root_file" custom attribute.

10. Restart the ORW application.

11. If ORW is configured to use OID, use the Permissions Management tool to assign permission grants to the secure custom work item.

Customizing the ORW Branding and Look and Feel

Changing the ORW Logo and Application Name

You may customize the ORW logo and application name. The logo and application name may be configured in `retail-workspace-page-config.xml`.

Changing the ORW Logo

You may change the ORW logo by installing a branding image in the images directory of the deployed ORW application, then adding a `<branding-uri>` element inside the `<retail-portal-page>` element. The text of the `<branding-uri>` element must specify the path to the logo image file.

When you change the branding image, you may also change the "branding-alt-text" attribute. The "alt" text is attached to the HTML `` tag that is used to render the branding image. Browsers display the "alt" text when the user hovers the mouse over the image.

Example: Changing the Branding Image and Specifying "alt" text for the Image In this example, you have created a new image that you would like to display instead of the default Oracle logo. The image is a GIF file named "myCompanyLogo.gif". You also want to specify "alt" text for the logo.

1. Copy `myCompanyLogo.gif` to the "images" directory of the ORW application (`$ORACLE_HOME/j2ee/<oc4j_instance>/applications/RetailWorkspace/images`). The URI (Uniform Resource Locator) for the image is "images/myCompanyLogo.gif".
2. Edit `retail-workspace-page-config.xml`. Locate the `<retail-portal-page>` element (at the beginning of the file). It will look similar to the following fragment of XML:

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Change the `retail-portal-page` element to include a `branding-alt-text` attribute to define the "alt" text. For this example, the "alt" text is "My Company":

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  branding-alt-text="My Company">
```

3. Insert a `<branding-uri>` element that specifies the path to `images/myCompany.gif` inside the `<retail-portal-page>` element:

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  branding-alt-text="My Company">
  <branding-uri>images/myCompanyLogo.gif</branding-uri>
```

Changing the Application Name

You may customize the title of the ORW application by defining a "branding-app-text" attribute in the `<retail-portal-page>` element of `retail-workspace-page-config.xml`. The "branding-app-text" is used to configure the text that occurs to the right of the branding image in the header of the ORW page.

You may also customize the title that is displayed in the browser's title bar by defining an "application-title" attribute in the <retail-portal-page> element.

Example: Changing the Application Name In this example, you have decided to change the application's title from "Retail Workspace" to "My Company's Retail Workspace". You want to use this title in both the application header, and in the browser title bar.

1. Edit `retail-workspace-page-config.xml`. Locate the <retail-portal-page> element (at the beginning of the file). It will look similar to the following fragment of XML:

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Change the retail-portal-page element to include a branding-app-text attribute to configure the text to display in the application header. Specify the value as "My Company's Retail Workspace":

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    branding-app-text="My Company's Retail Workspace">
```

2. Change the retail-portal-page element to include an application-title attribute to configure the text to display in the browser's title bar:

```
<retail-portal-page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    branding-app-text="My Company's Retail Workspace"
    application-title="My Company's Retail Workspace">
```

Creating a New ORW Skin

ORW allows you to change the appearance or look and feel of the application without having to rewrite any of the code that implements the application's user interface, by altering the application's skin.

A skin in ADF Faces is a global style sheet that only needs to be set in one place for the application. Instead of having to style each component, or having to insert a style sheet on each page, you can create one skin for the ORW application. Every component will automatically use the styles as described by the skin. Any changes to the skin will be picked up at runtime, so no change to code is needed. Skins are based on the Cascading Style Sheet specification, and use the CSS 3.0 syntax.

While it is possible to develop skins from scratch, it is recommended to use one of the two skins provided by the ORW application as a starting point, and then make the changes to colors, fonts and images to achieve the desired appearance.

The ORW default skin and images (`/skins/retailPortal`) should not be modified, but rather copied to a new directory and renamed to make the necessary customizations. However, the ORW application also provides a simple skin (`/skins/retailPortalSimple`) which can be copied and maintained in the same directory to quickly provide a different look and feel.

The following sections explain in more detail how to customize both the simple skin and the default skin.

Customizing the Simple Skin

This `retailPortalSimple` skin is intended to be a sample skin that can be copied and used to quickly and easily achieve a customized look and feel. The `retailPortalSimple` skin is located under `/skins/retailPortalSimple`. The `retailPortalSimple.css` file should not be modified directly. Instead, it is recommended to make a copy of the file, but keep it in the same directory location.

Note: The following steps for customizing the retailPortalSimple skin assume you do not need to make changes to the images used by the skin. If you intend to customize the skin images, you should follow the steps outlined in Customizing the Default Skin.

Perform the following steps to customize the simple skin:

1. Copy the retailPortalSimple.css and rename it within the same directory.
For example: /skins/retailPortalSimple/myCompanySimple.css
2. Modify the myCompanySimple.css file to specify any new colors, fonts, margins, padding, etc. as necessary to achieve the new look and feel (see Style Selectors Explained below).
3. Add a <skin> entry to the ORW adf-faces-skins.xml configuration file that defines your new copy of the simple skin.

The adf-faces-skins.xml defines the custom skins to be used in the application.

When adding the skin to the adf-faces-skins.xml configuration, the <skin> entry would look like this for the myCompanySimple skin:

```
<skin>
  <id>myCompanySimple</id>
  <family>myCompanySimple</family>
  <render-kit-id>oracle.adf.desktop</render-kit-id>
  <style-sheet-
name>skins/retailPortalSimple/myCompanySimple.css</style-sheet-name>
</skin>
```

For specific locales, the default retailPortal skin includes a separate skin for Japanese, Korean, or Chinese (zh_TW or zh_CN) locales. These are not provided for the retailPortalSimple skin, but they can easily be added by copying the myCompanySimple.css and adding the appropriate suffix for the supported locales. For example: myCompanySimple_xx.css files (where "xx" is "ja", "ko", "zh_TW" and "zh_CN".)

The font for the language specific locales can be set one time in the .AFDefaultFontFamily:alias. For example the myCompanySimple.css has the following font family:

```
.AFDefaultFontFamily:alias
{
  font-family:Tahoma, Verdana, Helvetica, sans-serif;
}
```

For Japanese, the font could be changed in the myCompanySimple_ja.css as such:

```
.AFDefaultFontFamily:alias
{
  font-family:'MS Gothic', Verdana, Helvetica, sans-serif;
}
```

If new locale specific skins are added, they must be defined in the adf-faces-skins.xml. For example, the Japanese version would look like this:

```
<skin>
  <id>myCompanySimple_ja</id>
  <family>myCompanySimple_ja</family>
  <render-kit-id>oracle.adf.desktop</render-kit-id>
```

```

    <style-sheet-name>skins/retailPortalSimple/myCompanySimple_
ja.css</style-sheet-name>
</skin>

```

4. Change the managed properties for the SkinsBean in the ORW faces-config.xml.

ORW utilizes a managed bean called SkinsBean to load the appropriate skin to use for the application. (For more information on the SkinsBean, see "[SkinsBean Explained](#)".) When adding a new skin, it is required to modify the values of the <managed-property> elements for the baseSkinFamily, baseStyleSheetName, and skinsDirectory values.

Below is an example of what the < managed-property > elements would look like if the myCompanySimple skin is being used:

```

<managed-bean>
  <managed-bean-name>skins</managed-bean-name>
  <managed-bean-class>oracle.retail.common.faces.bean.SkinsBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>baseSkinFamily</property-name>
    <property-class>java.lang.String</property-class>
    <value>myCompanySimple</value>
  </managed-property>
  <managed-property>
    <property-name>baseStyleSheetName</property-name>
    <property-class>java.lang.String</property-class>
    <value>myCompanySimple</value>
  </managed-property>
  <managed-property>
    <property-name>skinsDirectory</property-name>
    <property-class>java.lang.String</property-class>
    <value>skins/retailPortalSimple</value>
  </managed-property>
</managed-bean>

```

Customizing the Default Skin

This ORW default skin (located in /skins/retailPortal) should not be customized directly. To make customizations to this skin, the following steps should be followed:

1. Copy all directories and files of the /skins/retailPortal skin to a new directory tree, as a new sub-directory under the RetailWorkspace/skins directory

For example: /skins/myCompanySkins

2. Rename the retailPortal.css file.

For example: myCompany.css

3. Replace all of the URLs in the myCompany.css file that point to the skins/retailPortal/skin_images directory with the new directory.

For example: /skins/myCompanySkins/skin_images

4. Add a `<skin>` entry to the ORW `adf-faces-skins.xml` configuration file that defines your new skin; if Japanese, Korean and/or Chinese skin CSS files are being provided, add `<skin>` entries for each additional supported language.

The `adf-faces-skins.xml` defines the custom skins to be used in the application.

When adding the skin to the `adf-faces-skins.xml` configuration, the `<skin>` entry would look like this:

```
<skin>
  <id>myCompany</id>
  <family>myCompany</family>
  <render-kit-id>oracle.adf.desktop</render-kit-id>
  <style-sheet-name>skins/myCompanySkins/myCompany.css</style-sheet-name>
</skin>
```

Locale specific skins must also be defined here. For example, the Japanese version would look like this:

```
<skin>
  <id>myCompany_ja</id>
  <family>myCompany_ja</family>
  <render-kit-id>oracle.adf.desktop</render-kit-id>
  <style-sheet-name>skins/myCompanySkins/myCompany_ja.css</style-sheet-name>
</skin>
```

5. Change the base skin family, base style sheet name, and skins directory for the `SkinsBean` in the `faces-config.xml`.

ORW utilizes a managed bean called `SkinsBean` to load the appropriate skin to use for the application. (For more information on the `SkinsBean`, see "[SkinsBean Explained](#)".) When adding a new skin, it is required to modify the values of the `<managed-property>` elements for the `baseSkinFamily`, `baseStyleSheetName`, and `skinsDirectory` values.

Below is an example of what the `<managed-property>` elements would look like based on this approach:

```
<managed-bean>
  <managed-bean-name>skins</managed-bean-name>

  <managed-bean-class>oracle.retail.common.faces.bean.SkinsBean</managed-bean-class>

  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>baseSkinFamily</property-name>
    <property-class>java.lang.String</property-class>
    <value>myCompany</value>
  </managed-property>
  <managed-property>
    <property-name>baseStyleSheetName</property-name>
    <property-class>java.lang.String</property-class>
    <value>myCompany</value>
  </managed-property>
  <managed-property>
    <property-name>skinsDirectory</property-name>
    <property-class>java.lang.String</property-class>
    <value>skins/myCompanySkins</value>
  </managed-property>
</managed-bean>
```

Note: Before proceeding any further, it is recommended that you test the application at this point to ensure that nothing was missed in the conversion and configuration from retailPortal to myCompanySkins.

6. Modify the `myCompany.css` file to specify any new colors, fonts, margins, padding, etc. as necessary to achieve the new look and feel (see *Style Selectors Explained*). If the ORW application will be used in the Japanese, Korean, or Chinese (`zh_TW` or `zh_CN`) locales, make similar modifications to your `myCompany_xx.css` files (where "xx" is "ja", "ko", "zh_TW" and "zh_CN").
7. Modify or replace skin images in the `/skins/myCompanySkins/skin_images` directory, as necessary, to achieve the new look.

Style Selectors Explained

Style sheet rules encompass a style selector, which identifies an element, and a set of style definitions, which describe the element's appearance. The following example illustrates a style selector from the retailPortal default skin and definition that apply to the ShowDetailFrame core customizable component.

```
af|showDetailFrame::container
{
    border-style:none ;
    background-color:White;
    border-top:1px #979991 none;
    border-bottom:1px #979991 none;
    border-left:1px #979991 none;
    border-right:1px #979991 none;
}
```

This example defines styles for the main menu container of the ShowDetailFrame component. The style definition specifies menu background color, border-style and color of the menu's surrounding borders.

Oracle ADF Faces skins use the following three types of style selectors:

- Standard selectors—These directly represent an element that can have styles applied to it. For example `af|body` represents the `af:body` component. You can set CSS styles, properties, and icons for this element.
- Selectors with pseudo elements—Denotes a specific area of a component that can have styles applied. Pseudo elements are easily recognizable by a double colon followed by the portion of the component that the selector represents.

For example, `af|showDetailFrame::header-top-border` is the style selector for the top border of the header of a ShowDetailFrame component.

- Selectors that use the alias pseudo class—Used for a selector that sets styles for more than one component or more than one portion of a component.

For example, the `.retailPortalDefaultFontSize:alias` defines the default font size for the ORW application

SkinsBean Explained

The `SkinsBean` is an ORW managed bean that is responsible for retrieving the appropriate application skin based on locale.

The `getSkinFamily()` method uses the values of the `baseStyleSheetName` and `skinsDirectory` managed properties, and the browser locale to search the skins directory for a locale specific css file. If a locale specific file is found, then the skin

family name is constructed by appending an "_" (underscore) followed by the locale, to the value of the baseSkinFamily managed property and returned.

If a locale specific file is not found, then the getSkinFamily() method will simply return the value of the baseSkinFamily managed property.

Additional Skin Information and References

The ORW application only provides styling for those ADF Faces components utilized in the application. If new components are added, they need to be styled and defined in the .css file as well.

Note: Any modifications to add a custom skin need to be re-deployed each time ORW is installed. Oracle Retail recommends copying your custom skins to a backup directory before re-installing ORW to allow for re-applying the skin changes.

For more information on ADF Skins, please refer to the links below:

- ADF Faces skin selector documentation
<http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/doc/skin-selectors.html>
- Defining and Applying Styles to Core Customizable Components – Refer to the Oracle WebCenter Framework Developer’s Guide included in the Oracle WebCenter documentation.
- Developing and Using ADF Faces Skins
<http://www.oracle.com/technology/products/jdev/101/howtos/adfskins/index.html>

Dashboard Development Tutorial

Introduction

This tutorial teaches you how to create a dashboard and make it look and feel like the demonstration dashboards supplied with the ORW application. The lessons in this tutorial will cover the following topics:

- How to develop dashboard content for display in the ORW application.
- How to create a dashboard application in JDeveloper using JSF, ADF Faces, and Oracle WebCenter components.
- How to secure the dashboard.
- How to deploy the dashboard.
- How to configure ORW to display the dashboard.

Note that this document uses the MyDashboard application as an example.

Note: The procedures in this guide do not contain steps to save your work. Be sure to save your work often by clicking the double floppy disk icon in JDeveloper.

Audience

This tutorial is intended for content developers and administrators who need to deploy content developed for ORW. You are expected to have some familiarity with JDeveloper, Oracle Application Development Framework (ADF), Oracle ADF Faces, and Java.

Getting Started

This section tells you how to download the correct version of Oracle JDeveloper to complete the steps in this tutorial.

Downloading Oracle JDeveloper 10.1.3.4 with the Web Center Extensions

You can download Oracle JDeveloper (10.1.3.4) from the Oracle Technology Network:

<http://www.oracle.com/technology/products/jdev/index.html>

1. Select the **download** link.
2. On the Oracle JDeveloper Downloads page, click on the Oracle JDeveloper (10.1.3.4) link.

3. Download and unzip to a directory of your choice.

Creating the Dashboard Application and Registering the Portlet Producer

In this lesson, you will learn how to create a new JDeveloper application and register a portlet producer.

Create a New JDeveloper Application

ORW supports only Internet Explorer, version 6 and later. By default, JDeveloper launches the user's default browser, but it is possible to alter JDeveloper preferences to specify the browser.

1. In the Applications Navigator pane, select the project that includes the dashboard page.
2. Open the **Tools** menu and select the **Preferences** menu item.
3. In the Preferences dialog, select **Web Browser and Proxy**.
4. Fill in the path to the Internet Explorer application in the Browse Command Line field or select **Browse** to search for the browser executable.
5. Once the browser command line has been configured, click **OK** to save the changes.

To create the JDeveloper application:

1. Open JDeveloper by double-clicking `jdev.exe` in the directory where JDeveloper resides.
2. Select the **Applications Navigator** view tab.
3. In the Applications Navigator, highlight **Applications** and right-click.
4. Select **New Application** from the menu.
5. Enter the details relevant to the dashboard you are developing into the wizard.
 - a. Name your application **MyDashboard**.
 - b. For the Application Template, select **WebCenter Application** from the menu.
 - c. Accept the default directory location and make a note of it.

When you choose the WebCenter Application template when creating the application, JDeveloper creates the following three projects by default in your application:

- Model
- Portlets
- ViewController

You may later find that your application does not require some of the projects, for example, if you do not create portlets or you do not employ JDeveloper features that use the Model project. The ViewController project folder is the only folder needed for the MyDashboard application. There are two different ways you can remove unnecessary projects:

- Erase a project from the disk by selecting the project in the Applications Navigator. Select **File** and then **Erase from Disk**.

- Delete the project by right-clicking the project folder and selecting **Delete**. The Delete operation only removes the project from the application navigator list. It does not delete the project from the file system.

Sometimes JDeveloper does not do a clean remove. In these cases, you need to delete these folders manually in file explorer. The only way in which to tell if the clean remove failed is to examine the directories in the file system.

Verify Access to the Deployed Retail Workspace Portlet Producer

Before proceeding with the registration process, make sure that ORW portlets have been deployed and you have the URL of their deployment. See step 8 in "[Register Portlet Producer](#)" for URL information.

You can confirm the deployment of the portlet by entering the WSRP2 WSDL URL in a browser. When ORW was installed, the URL for the portlets WSDL should have been noted.

```
http://<host>:<port>/<context-root>
```

The following example shows the Retail Portlets WSDL.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:bind="urn:oasis:names:tc:wsrp:v2:bind"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:oasis:names:tc:wsrp:v2:wsdl">
  <import namespace="urn:oasis:names:tc:wsrp:v2:bind" location="wsrp_v2_
bindings.wsdl" />
- <wsdl:service name="WSRP_v2_Service">
- <wsdl:port binding="bind:WSRP_v2_ServiceDescription_Binding_SOAP" name="WSRP_v2_
ServiceDescription_Service">
  <soap:address location="http://<host>:<port>/RetailPortlets/portlets/WSRP_v2_
ServiceDescription_Service" />
</wsdl:port>
- <wsdl:port binding="bind:WSRP_v2_Markup_Binding_SOAP" name="WSRP_v2_Markup_
Service">
  <soap:address
location="http://http://<host>:<port>/RetailPortlets/portlets/WSRP_v2_Markup_
Service" />
</wsdl:port>
- <wsdl:port binding="bind:WSRP_v2_Registration_Binding_SOAP" name="WSRP_v2_
Registration_Service">
  <soap:address
location="http://http://<host>:<port>/RetailPortlets/portlets/WSRP_v2_
Registration_Service" />
</wsdl:port>
- <wsdl:port binding="bind:WSRP_v2_PortletManagement_Binding_SOAP" name="WSRP_v2_
PortletManagement_Service">
  <soap:address
location="http://http://<host>:<port>/RetailPortlets/portlets/WSRP_v2_
PortletManagement_Service" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Register Portlet Producer

ORW provides five types of portlets dashboard developers can utilize.

- ORWOBIEEAlertsPortlet—Used for displaying Oracle BI EE Alerts.
- ReportPortlet—Used for displaying a Report URL.
- RSSPortlet—Used for displaying the contents of a RSS Feed such as http://www.oracle.com/technology/products/jdev/jdeveloper_news.xml
- URLPortlet—Used for displaying a URL link such as <http://www.oracle.com>
- ORWSlideshowPortlet—Used for displaying up to 8 configured URL links in a slide show format

Registering a portlet producer makes these portlets available to your dashboard for consumption. JDeveloper provides options for registering Oracle proprietary PDK portlets or WSRP standard portlets. Oracle Retail portlets are JSR 168 portlets that use the optional Oracle WSRP 2 extensions for inter-portlet communication.

Use the following procedure to register a WSRP portlet producer. Note that the producer registration requires not only that the ORW portlet producer already be deployed, but that the portlet producer is running as well.

1. In the Applications Navigator, right-click the ViewController project and select **New** from the context menu.
2. In New Gallery, filter by **All Technologies** then under Categories, expand the Web Tier node and select Portlets.
3. In New Gallery under Items, select **WSRP Producer Registration**.
4. Click **OK**.
5. If the Welcome page appears, click **Next**.
6. In the Name field, enter a name for the producer.
7. Click **Next**.
8. In the URL Endpoint field, enter the producer's URL.

Example:

`http://<host>:<port>/<context-root>/portlets/wsrp2?WSDL`

where:

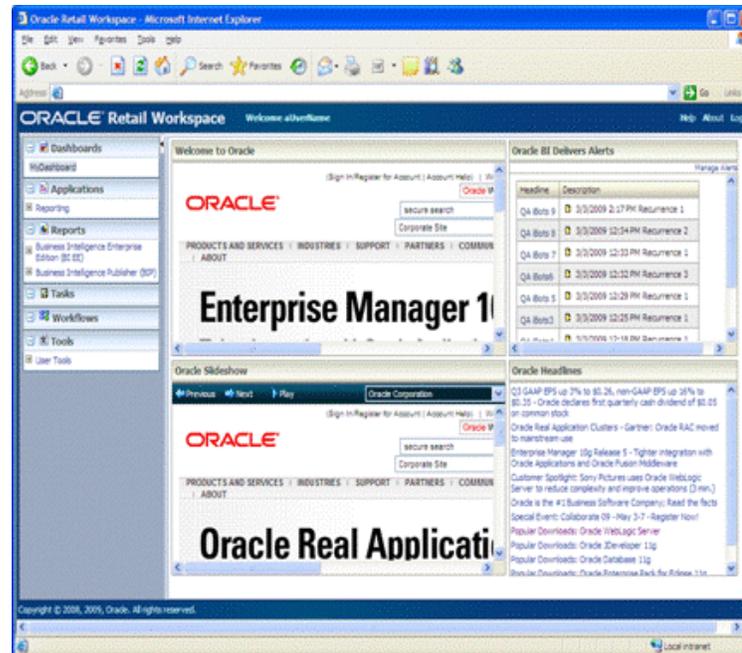
- host is the server to which your producer has been deployed.
 - port is the HTTP Listener port number.
 - context-root is the context root of the deployed ORW portlets. By default, the ORW portlet producer application context roots are:
 - RetailPortlets – includes the URLPortlet, RSSPortlet, ReportPortlet and ORWOBIEEAlertsPortlet
 - RetailSlideshowPortlet – includes the ORWSlideshowPortlet
 - portlets/wsrp2?WSDL is static text.
9. Accept defaults for other settings.
 10. To complete registration of the WSRP portlet producer, click **Finish**.
This creates a Portlet Producers folder under your dashboard application.

Planning your Dashboard

Before developing your dashboard page, it is a good idea to visualize what your dashboard will look like. Making a screen mockup is helpful.

In following sections you will be creating a dashboard page named **MyDashboard**. The MyDashboard page will be developed using the ORWSlideshow Portlet, Report portlet, RSS portlet and ORWOBIEEAlerts Portlet, and will look similar to the following example:

Figure 9–1 Example of a Dashboard with Multiple Portlets



Note: This tutorial uses the ORWOBIEEAlerts Portlet and assumes that the ORWOBIEEAlerts Application has been deployed on the Oracle BI EE Server with iBots enabled. If you do not have this functionality set up, ignore the tutorial portions that mention ORWOBIEEAlerts Portlet or use a URL or Report Portlet in its place.

You will need the following information to use the ORWOBIEEAlerts Portlet:

- The URL of the ORWOBIEEAlerts application. This uses the following form:


```
http://<host>:<port>/<orwobieealerts-context-root>/faces/ORWOBIEEAlertsDisplay.jspx
```

 - <host> is the machine where ORWOBIEEAlerts is deployed
 - <port> is the port of the machine where ORWOBIEEAlerts is deployed.
 - <orwobieealerts-context-root> is the context-root of the ORWOBIEEAlerts Application deployment.
- The Oracle BI EE deployment Context Root.
- The Oracle BI EE Manage Alerts URL. This uses the following form:

```
http://<host>:<port>/<analytics-context-root>/saw.dll?Alerts
```

where:

- <host> is the machine where Oracle BI EE is deployed
- <port> is the port of the machine where Oracle BI EE is deployed.
- <analytics-context-root> is the Context Root for the Oracle BI EE Delivers deployment.

Contact your system administrator for the values of <host>, <port>, and <analytics-context-root>. For information on how to integrate Oracle BI EE Alerts with ORW see [Chapter 6, "Integration Methods and Communication Flow"](#).

In the figure, the MyDashboard dashboard is displayed in the large content area of the ORW application. Notice the dashboard consists of four separate sub-panes each with their own content. These sub-panes are portlets.

The top-left portlet, titled **Welcome to Oracle**, is the Report Portlet. The Report portlet is typically used to display an Oracle BI EE or BIP report URL. In our example it is displaying a web page - the Oracle web site.

The top-right portlet, titled **Oracle BI Delivers Alerts**, is the OBIEEAlerts Portlet. The OBIEEAlerts portlet is typically used to display a list of active Oracle BI EE Alerts for the user.

The bottom-left portlet, is the ORW Slideshow Portlet. The ORW Slideshow portlet is typically used to display up to eight configured URLs in a slideshow format. In our example it is displaying the Oracle Corporate, Oracle Retail, and Oracle Technology Network Web sites.

Note: The URL portlet is similar to the Reports portlet in that it displays a URL in an HTML <iframe>. Use the Reports portlet for displaying Oracle Business Intelligence Enterprise Edition (BI EE) and Business Intelligence Publisher (BIP) report URLs. Use the URL portlet to display other web pages. Use the ORWSlideshowPortlet to display up to eight URLs in a slideshow format.

The bottom right portlet, titled **Oracle Headlines**, is an RSS portlet. You will use the RSS portlet in a dashboard to display the contents of an RSS feed.

Dashboard Layout

We will lay out our dashboard with 2 rows of portlets, with 2 portlets in each row. When laying out the page, we will use the WebCenter PanelCustomizable component as a container to hold the portlets. We will start with one outer PanelCustomizable component with a vertical layout. Inside the outer PanelCustomizable, we will stack two PanelCustomizable components with horizontal layouts. Inside each of these inner panels we will place two portlets.

Now that we have planned our layout, we can begin developing the dashboard.

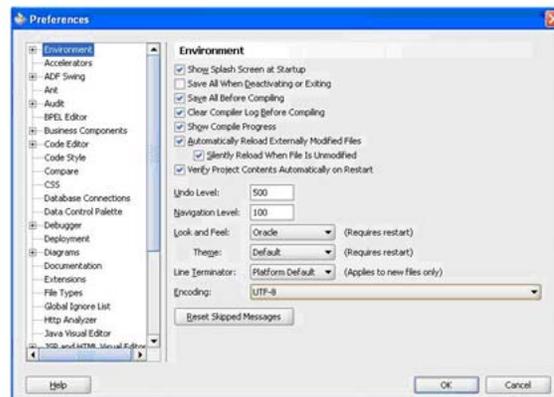
Creating the Dashboard

In this lesson you will create the MyDashboard page.

Create the Dashboard Page (MyDashboard.jspx)

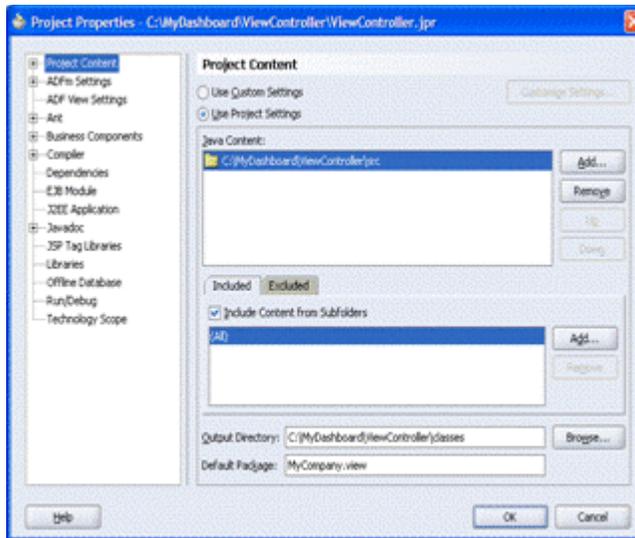
1. The dashboard you are about to create should use UTF-8 character encoding. Before creating your dashboard page, you may wish to set the JDeveloper default for character encoding to UTF-8.
 - a. Open the **Tools** menu and select the **Preferences** menu item.
 - b. In the Preferences dialog, select **Environment**.
 - c. Open the **Encoding** menu and select **UTF-8**.
 - d. To save your preferences and close the dialog, click **OK**.

Figure 9–2 Preference Dialog Box



2. The default 'view' package should be changed in the ViewController project so that the page definitions will automatically get placed below our defined package. Fully qualified page definitions need to be unique because later on we will be assigning authorization to customize, personalize and view a page based on the page definition name. If all page definitions are created under the default 'view' package, there is a risk of name collisions with page definitions in other applications that have similarly named pages.
 - a. In the Application Navigator, right-click on the ViewController project for the MyDashboard application and select Project Properties from the context menu.
 - b. Select **Project Content**.
 - c. Change the Default Package to MyCompany.view.
 - d. To save your changes, click **OK**.

Figure 9–3 Project Properties Dialog Box



3. In the Applications Navigator, right-click the ViewController for the MyDashboard application and select **New** from the context menu.
4. In New Gallery under Categories, expand the Web Tier node and select **JSF**.
5. In New Gallery under Items, select **JSF JSP**.
6. To open the Create JSF JSP wizard, click **OK**.
7. If the Welcome page appears, click **Next**.
8. In the File Name field, enter the name for the dashboard: MyDashboard.jspx.
9. Leave the Directory Name field as the default.
10. In the Type field, select JSP Document (*.jspx).

Note: Create WebCenter application pages as JSP documents (JSPX) rather than JSP pages (jsp).

11. Click **Next**. On step 2, Component Binding, ensure that the **Do Not Automatically Expose UI Components in a Managed Bean** checkbox is checked.
12. Click **Next** until the Tag Libraries screen is displayed. Select **Project Technologies** in the **Filter By** box. Make sure the following libraries are moved to the Selected Libraries list:
 - ADF Faces Components
 - ADF Faces HTML
 - ADF Portlet Components
This library is required when you plan to place portlets on application pages.
 - Customizable Components Core
This library is required when you plan to use the WebCenter Core Customizable Components, PanelCustomizable and ShowDetailFrame.
 - JSF Core

- JSF HTML

13. Click **Finish**. The wizard closes. JDeveloper then creates the `MyDashboard.jspx` file and open it in an editor pane.
14. In the editor pane for `MyDashboard.jspx`, click on the **Source** tab to go to the source of the `jspx` page. On the first line, set `encoding='UTF-8'` if it is not already set.

Make sure encoding or charset is set to UTF-8 wherever specified. Note that you should not have to do this if you have already set UTF-8 as the default encoding.

Look for the following tags that may have encoding or charset values:

- `<?xml version='1.0' encoding='UTF-8'?>`
- `<jsp:directive.page contentType="text/html;charset=UTF-8"/>`
- `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>`

Layout the Dashboard

You now are ready to lay out the dashboard page. You will use `PanelCustomizable` and `ObjectSpacer` components to control the layout of your page.

Note: ADF Faces and JSF provide several other components that can be used for controlling page layouts.

1. Add the outer `panelCustomizable` container to the page.
 - a. Right-click on the **h:form** in `MyDashboard.jspx`'s Structure pane.
 - b. In the context menu, open the **Insert inside h:form** menu, and click **Browse**.
 - c. Select **Customizable Components Core** from the menu at the top of the dialog.
 - d. Select `Panel Customizable` in the bottom window and click **OK**.
2. Select the outer `cust:panelCustomizable` in the Structure pane. In the Property Inspector for the `PanelCustomizable`, scroll to the `Layout` property and set its value to `Vertical`.
3. Nest two `panelCustomizable` containers inside the outer `panelCustomizable`. To do this:
 - a. In the Structure pane, right-click on the outer `cust:panelCustomizable`.
 - b. Open the "Insert inside `cust:panelCustomizable`" menu item, and follow the same procedure you used in step 1, navigating through the sub-menus to insert the `Panel Customizable`.
 - c. Select the outer `panelCustomizable` in the Structure pane again and repeat this procedure to create the second nested `panelCustomizable`.
4. Change the layout of the two inner `panelCustomizable` containers to `horizontal`. This can be done by selecting each `cust:panelCustomizable` in the Structure pane and changing the value of the `Layout` property in the Property Inspector to `horizontal`. Make sure that the outer `panelCustomizable`'s layout is `vertical`.
5. The look of your dashboard will be improved if there is some space between the portlets. One way to control spacing between components is to insert an ADF Faces `ObjectSpacer` component between them. Insert an `ObjectSpacer` between the

inner panelCustomizable components to add some space between the two rows of portlets.

- In the Structure pane, select the first of the two inner panelCustomizable components. Right-click to open the context menu.
- In the context menu, mouse over **Insert after cust:panelCustomizable** to open the menu.
- In the menu, select **ADF Faces Core**.
- In the Insert ADF Faces Core Item dialog, select **ObjectSpacer** and click **OK**. An af:objectSpacer element will be inserted in your page after the cust:panelCustomizable.
- In the ObjectSpacer Property Inspector, change the Width or Height as appropriate to provide spacing. Since the two inner panelCustomizable components are layed out vertically, set the Height to 4 to control the vertical space between the panels.

Set PanelCustomizable Properties

The PanelCustomizable component has several optional attributes in addition to the Layout attribute. For example, it is possible to display a header bar at the top of the panel. It is also possible to enable an action menu that allows you to maximize and minimize the container, and so on. For this dashboard, you will only be using the PanelCustomizable as a layout container. In the following step, you will set attributes to control the behavior of the PanelCustomizable.

- Repeat this step for each cust:panelCustomizable tag:
Select the cust:panelCustomizable in the MyDashboard.jspx Structure pane. Use the Property Inspector to set the following property values:

General Category:

- DisplayHeader = false
- DisplayScrollbar = auto

Actions Category:

- IsEditable = false
- IsHelpAvailable = false
- IsMaximizable = false
- IsMinimizable = false
- IsMoveable = false
- IsSeededInteractionAvailable = false
- IsShowContentEnabled = true

Core Category:

- Rendered = true

Add Portlets to the Dashboard

Now that you have established the page layout, you can begin placing portlets in the individual panelCustomizables. You will add two portlets to each of the two inner PanelCustomizable components.

Note: When adding a portlet, make sure you are adding the portlet to one of the inner `PanelCustomizable` components. The outer `PanelCustomizable` is used merely to contain and layout the inner panels.

In "[Planning your Dashboard](#)", it was determined that you will be adding four portlets to the dashboard. In the following steps, you will add a Reports Portlet and an OBIEEAlerts Portlet to the top row of the layout (the first of the two inner `PanelCustomizable` components). You then will add an ORW Slideshow Portlet and an RSS Portlet to the bottom row of the layout (the second of the two inner `PanelCustomizable` components).

Note: This is just one variation of portlets that could be used on this page. You can substitute the URL Portlet for the other portlets, or add and remove portlets as desired.

1. In the `MyDashboard.jspx` Structure pane, expand the outer-most `cust:panelCustomizable`. Locate the first of the two inner `cust:panelCustomizable` components. Right click on this `cust:panelCustomizable` in the Structure pane. In the context menu, open **Insert inside cust:panelCustomizable** and then open the **ADF Portlet Components** menu. Another menu should open up that includes the following portlets that are included in the ORW portlet producer, which you registered in a previous section:
 - `ORWOBIEEAlertsPortlet`
 - `ReportPortlet`
 - `RSSPortlet`
 - `ORW Slideshow Portlet`
 - `URLPortlet`
2. Select **ReportPortlet**. After a little time to process the menu selection, JDeveloper will insert an `adfp:portlet` tag into `MyDashboard.jspx`. Behind the scenes, JDeveloper will also generate some required metadata files that describe the portlet you just added to the page.

Note: JDeveloper creates bindings and metadata for portlets when they are added to a WebCenter application page. JDeveloper associates these bindings and metadata by automatically generating IDs and portlet instance names. It is not advisable to change portlet or portlet variable names that have been generated by JDeveloper.

3. In the Property Inspector pane for the `adfp:portlet` that you just added, set the following properties to the following values and let other properties default:

Note: The ReportPortlet, RSSPortlet, ORW Slideshow Portlet, and URL Portlet have the ability to be customized or personalized. Since a security model is not in place at this point to support these options, the IsCustomizeModeAvailable and IsPersonalizeModeAvailable properties will be set to false here. Later, after the application is secured (in the section [Adding Customization and Personalization](#)), these properties will be set to true. You will then test the customization and personalization using the ORW Slideshow Portlet as an example.

Actions category:

- IsAboutModeAvailable = false
- IsConfigModeAvailable = false
- IsCustomizeModeAvailable = false
- IsDetailModeAvailable = false
- IsHelpModeAvailable = false
- IsLinkModeAvailable = false
- IsMaximizable = false
- IsMinimizable = false
- IsMovable = false
- IsNormalModeAvailable = false
- IsPersonalizeModeAvailable = false
- IsPreviewModeAvailable = false
- IsPrintModeAvailable = false
- IsSeededInteractionAvailable = false

Core category:

- Rendered = true

Display Options category:

- Background = light
- DisplayScrollbar = auto
- DisplayHeader = true
- RenderPortletInIFrame = false

4. In the Text property under the General category in the Property Inspector, enter the title of the portlet. For this example, enter "Welcome to Oracle".

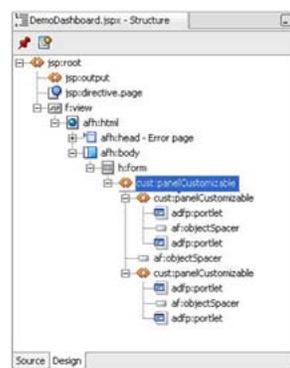
Note: In following sections, you will learn how to internationalize the title. You will also learn how you can modify your dashboard to pass the title as a parameter to the dashboard.

5. Add an OBIEEAlertsPortlet as the second portlet in the first row:

- a. In the Structure pane, select the `adfp:portlet` you just added (the ReportPortlet).
 - b. Right click on this `adfp:portlet`.
 - c. In the context menu, open **Insert after adfp:portlet** and then open the **ADF Portlet Components** menu.
 - d. Select **ORWOBIEEAlertsPortlet**. JDeveloper will insert another `adfp:portlet` tag into `MyDashboard.jspx` and will also generate some required metadata files that describes the OBIEEAlerts portlet you just added to the page.
6. Select the new ORWOBIEEAlertsPortlet in the Structure pane. In the Portlet Property Inspector, set the property values recommended above in step 3. It is not recommended to set the `text` attribute of the ORWOBIEEAlertsPortlet, as the portlet title is defaulted to Oracle BI Delivers Alerts.
 7. In the Structure pane, select the first of the two portlets you just added (the ReportPortlet), and then use the appropriate JDeveloper context menus to insert an ObjectSpacer after the portlet. Since the parent container of the portlets is laid out horizontally, change the Width property to 4 to control the horizontal spacing between the two portlets.
 8. Following the procedures you have just learned, add an ORW Slideshow Portlet and an RSSPortlet to the bottom row of the layout, that is, to the second of the two inner `panelCustomizable` components. It is not necessary to set the `text` attribute of the RSSPortlet, as the portlet will get its title from the RSS feed.

At this point, your dashboard structure pane will look similar to the following figure.

Figure 9–4 Dashboard Structure Pane



Layout Note: For more examples of how to lay out dashboards, look at the example dashboards that were packaged with ORW.

The example dashboards were created for a 1024 x 768 screen resolution.

The example dashboards use `objectSpacers` (an ADF faces Core Component) to space the components. The size of the spacers between components is 4. You can use `af:objectSpacer` to provide spacing if you want your dashboards to look similar to the ORW examples.

Set Portlet Parameters

You now have a dashboard with four portlets. Next, you will make some changes to the page's metadata to configure some values required to display data in the OBIEEAlerts, ORWSlideshow, Report and RSS portlets.

1. Right click on the `jspx` page in the Application Navigation pane and click **Go to Page Definition**. The `MyDashboardPageDef.xml` file will open in an editor pane and the structure pane.

Note: JDeveloper generates a PageDef's file name consisting of the `jspx` page name followed by `PageDef.xml`.

2. In the Structure pane, expand the **executables** node. Underneath executables, you will see a **variables** node, and four more nodes, one for each portlet you added to the page. The portlet nodes have been automatically named by JDeveloper.
3. Expand the first `ReportPortlet` node, and then expand its **parameters** node. The following parameters will be listed:
 - `PORTLET_TITLE`
 - `URL_TO_SHOW`
 - `PORTLET_WIDTH`
 - `PORTLET_HEIGHT`
 - `ALT_URL`
 - `ALT_URL_LABEL`
4. Expand the `ORWOBIEEAlertsPortlet` node, and then expand its parameters node. The following parameters will be listed:
 - `ORW_OBIEE_ALERTS_APPLICATION_URL`
 - `OBIEE_MANAGE_ALERTS_URL`
 - `OBIEE_CONTEXT_ROOT`
 - `PORTLET_WIDTH`
 - `PORTLET_HEIGHT`
5. Expand the `ORWSlideshowPortlet` node, and then expand its **parameters** node. The following parameters will be listed:
 - `PORTLET_WIDTH`
 - `PORTLET_HEIGHT`
 - `PORTLET_TITLE`
 - `SLIDESHOW_ON`
 - `SLIDE_SPEED`
 - `TRANSITION_TIME`
 - `TRANSITION_EFFECT`
 - `OPEN_IN_NEW_WINDOW_LINK_RENDERED`
 - `OPEN_IN_NEW_WINDOW_LINK_TEXT`
 - `SLIDE1_URL`

- SLIDE2_URL
 - SLIDE3_URL
 - SLIDE4_URL
 - SLIDE5_URL
 - SLIDE6_URL
 - SLIDE7_URL
 - SLIDE8_URL
 - SLIDE1_ALT_URL
 - SLIDE2_ALT_URL
 - SLIDE3_ALT_URL
 - SLIDE4_ALT_URL
 - SLIDE5_ALT_URL
 - SLIDE6_ALT_URL
 - SLIDE7_ALT_URL
 - SLIDE8_ALT_URL
 - SLIDE1_TITLE
 - SLIDE2_TITLE
 - SLIDE3_TITLE
 - SLIDE4_TITLE
 - SLIDE5_TITLE
 - SLIDE6_TITLE
 - SLIDE7_TITLE
 - SLIDE8_TITLE
6. Expand the RSSPortlet node, and then expand its **parameters** node. The following parameters will be listed:
- RSS_URL
 - PORTLET_WIDTH
 - PORTLET_HEIGHT
 - SHOW_ITEM_DESCRIPTION
7. In the Structure pane, expand the **variables** node. The child elements of the **variables** node are variable elements that expose the portlet parameters for each portlet instance on the page.

Since you added an RSS Portlet to the page, you will see variables for the RSSPortlet. If JDeveloper named the first RSSPortlet instance as RSSPortlet1_1, you will see the following variables: RSSPortlet1_1_RSS_URL, RSSPortlet1_1_PORTLET_WIDTH, RSSPortlet1_1_PORTLET_HEIGHT, and RSSPortlet1_1_SHOW_ITEM_DESCRIPTION. You will also see variables for the ReportPortlet, ORWSlideshowPortlet and ORWOBIEEAlertsPortlet instances you added to the page.

Note: JDeveloper automatically generates the IDs for the portlet instances, and automatically generates the variable names by concatenating the portlet parameter names to the portlet ID.

8. In the Structure pane, locate the PORTLET_TITLE variable for the top left ReportPortlet. It will be named something like ReportPortletX_Y_PORTLET_TITLE (where "X" and "Y" are instance numbers generated by JDeveloper). Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter a value. For our Example, enter Welcome to Oracle. Locate the URL_TO_SHOW variable and enter a URL value. For our example, enter `http://www.oracle.com`. This is the URL that will be displayed in the portlet. In an ORW production dashboard, you will be setting this to the URL of a BI EE or BIP report.
9. Select the PORTLET_WIDTH variable for the top left ReportPortlet. In the property inspector, set the DefaultValue property to 475. Select the PORTLET_HEIGHT variable for this portlet and set the DefaultValue to 230 in the property inspector.
10. In the Structure pane, locate the ORW_OBIEE_ALERTS_APPLICATION_URL variable for the ORWOBIEEAlertsPortlet. It will be named something like ORWOBIEEAlertsPortletX_Y_ORW_OBIEE_ALERTS_APPLICATION_URL (where "X" and "Y" are instance numbers generated by JDeveloper). Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter the URL for the ORWOBIEEAlerts Application. Enter `http://<host>:<port>/<orwobieealerts-context-root>/faces/ORWOBIEEAlertsDisplay.jspx`, where <host>, <port> and <orwobieealerts-context-root> comprise the ORWOBIEEAlerts application URL. The ORWOBIEEAlerts Application URL parses Oracle BI EE Alerts and displays them in a table, which will be displayed in the portlet.
11. In the Structure pane, locate the OBIEE_MANAGE_ALERTS_URL variable for the ORWOBIEEAlertsPortlet. It will be named something like ORWOBIEEAlertsPortletX_Y_OBIEE_MANAGE_ALERTS_URL. Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter the URL for the Oracle BI EE Alerts. It is of the form,


```
http://<host>:<port>/<analytics-context-root>/saw.dll?Alerts
```

 where <host> and <port> are the host and port numbers where Oracle BI EE presentation services are deployed and <analytics-context-root> is the context-root of Oracle BI EE presentation services.
12. Locate the OBIEE_CONTEXT_ROOT variable for the ORWOBIEEAlertsPortlet. It will be named something like ORWOBIEEAlertsPortletX_Y_OBIEE_CONTEXT_ROOT. Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter the context root for the ORWOBIEEAlerts Application. For our example, enter <analytics-context-root>.

Note: If your Oracle BI EE Answers application is deployed to analytics context root you do not need to set the OBIEE_CONTEXT_ROOT variable as it defaults to **analytics**.

13. Select the PORTLET_WIDTH variable for the ORWOBIEEAlertsPortlet. In the property inspector, set the DefaultValue property to 325. Select the PORTLET_

HEIGHT variable for this portlet and set the DefaultValue to 217 in the property inspector.

14. In the Structure pane, locate the PORTLET_WIDTH variable for the ORWSlideshowPortlet. It will be named something like ORWSlideshowX_Y_PORTLET_WIDTH (where "X" and "Y" are instance numbers generated by JDeveloper).
15. Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter a numeric value for the width. For our example, enter 475. This is the width of the portlet that will be displayed.
16. Select the PORTLET_HEIGHT variable. In the property inspector, locate the DefaultValue property for this portlet and set the DefaultValue to 230.
17. Select the PORTLET_TITLE variable. In the property inspector, set the DefaultValue to Oracle Slideshow.
18. Select the SLIDE_SPEED variable. In the property inspector, set the DefaultValue property variable to 5. This value is the amount of time in seconds, that one slide is displayed in the portlet before it transitions to the next configured slide while in the play mode.
19. Select the SLIDE1_URL variable. In the property inspector, set the DefaultValue to `http://www.oracle.com`.
20. Select the SLIDE2_URL variable. In the property inspector, set the DefaultValue to `http://www.oracle.com/industries/retail`.
21. Select the SLIDE3_URL variable. In the property inspector, set the DefaultValue to `http://www.oracle.com/industries/high_tech`.

Note: You can experiment and specify URLs for different Web pages. However, the page must be one that is able to render correctly within an HTML `<iframe>`.

22. Select the SLIDE1_TITLE variable. In the property inspector, set the DefaultValue to Oracle Corporation.
23. Select the SLIDE2_TITLE variable. In the property inspector, set the DefaultValue to Oracle Retail.
24. Select the SLIDE3_TITLE variable. In the property inspector set the DefaultValue to Oracle Technology.

Note: In this example we have chosen not to set an alternative URL for the three slide URLs. By default, the ORWSlideshow will use the slide URL as the alternate URL if one is not specified. You can add alternate URLs for the SLIDE1_ALT_URL, SLIDE2_ALT_URL, and SLIDE3_ALT_URL variables.

In order to launch these alternate URLs, you will need to set the OPEN_IN_NEW_WINDOW_LINK_RENDERED variable to true.

25. In the Structure pane, locate the RSS_URL variable for the RSSPortlet. It will be named something like RSSPortletX_Y_RSS_URL (where "X" and "Y" are instance numbers generated by JDeveloper). Click on this node. In the property inspector for this variable, locate the DefaultValue property and enter a URL for an RSS

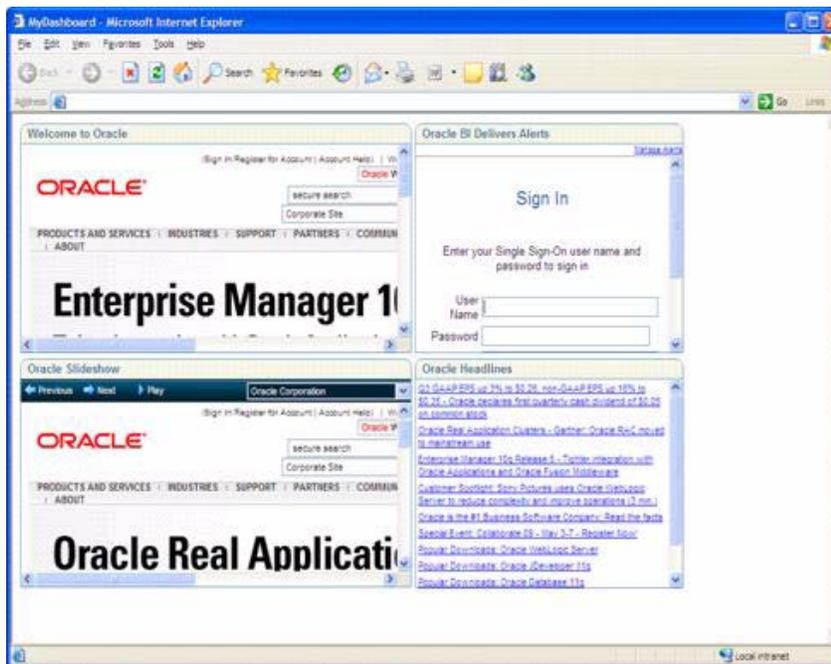
feed. For our example, enter `http://www.oracle.com/rss/rss_ocom_corpnews.xml`. This is the RSS feed that will be displayed in the portlet.

Note: The RSSPortlet supports RSS feeds that adhere to the RSS version 2.0 specification.

26. Select the PORTLET_WIDTH variable for the RSSPortlet. In the property inspector, set the DefaultValue property to 325. Select the PORTLET_HEIGHT variable for this portlet and set the DefaultValue to 230 in the property inspector.
27. Test the new Dashboard in JDeveloper.

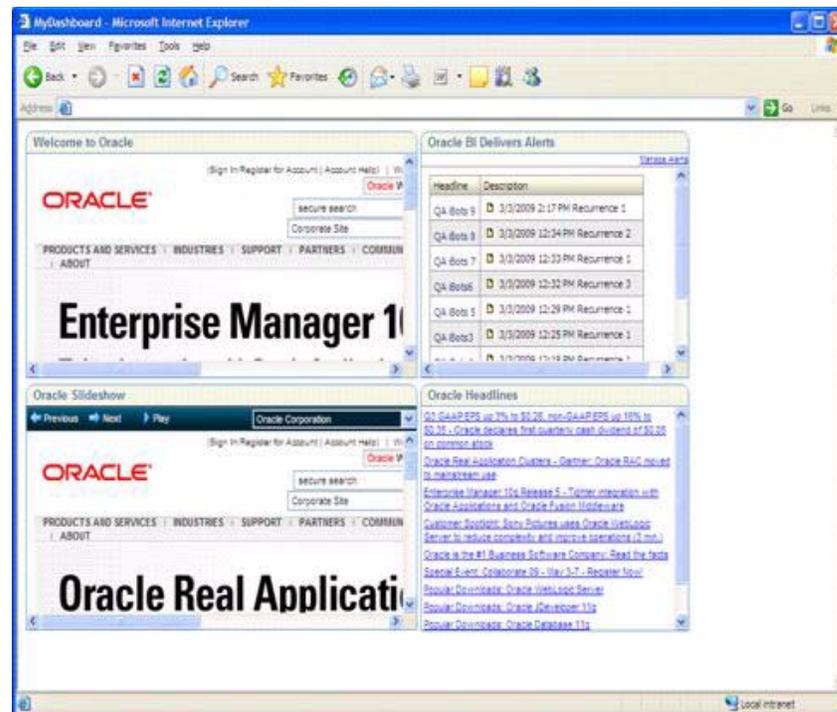
Right-click on the MyDashboard.jspx page in the Application Navigation pane and click **Run** in the context menu. This launches the dashboard in a browser window. You will notice that OBIEEAlerts Portlet is requiring you to login into the SSO server. This is because the dashboard has not been setup for security yet. Once the dashboard is SSO enabled and configured to display inside Retail Workspace it will not prompt you for login credentials. Once you input the user id and password, the OBIEEAlerts Portlet will present the list of Alerts for your user if any are available. See [Chapter 6, "Integration Methods and Communication Flow"](#) for information on how to integrate ORW with Oracle BI EE Alerts.

Figure 9–5 Dashboard in a Browser Window, Before Logging On to SSO



Dashboard after logging into SSO server for the Alerts Portlet.

Figure 9–6 Dashboard After Login On to SSO



Summary of Portlet Parameters

Table 9–1 Report and URL Portlet Input Parameters

Parameter	Description
PORTLET_TITLE	This is an optional parameter. The Report and URL titles default to ReportPortlet and URLPortlet respectively. If this parameter is set, it allows the dashboard developer to override the default title of the portlet.
URL_TO_SHOW	This is the URL the portlet will display in its content space. It will be displayed as a source of an IFRAME. There is no default URL if one is not specified.
PORTLET_WIDTH	This is an optional parameter. The URL portlet width defaults to 400. If this parameter is set, it allows the dashboard developer to override the default width of the portlet.
PORTLET_HEIGHT	This is an optional parameter. The URL portlet height defaults to 250. If this parameter is set, it allows the dashboard developer to override the default height of the portlet.
ALT_URL	This is an optional parameter. If specified, the portlet will have a hyperlink visible on the top-right that launches the specified URL in a separate window. The hyperlink label defaults to "Open in New Window" for a URL Portlet and "Show Details" for a Report Portlet. This can be overridden by setting ALT_URL_LABEL parameter.
ALT_URL_LABEL	This is an optional parameter. This is the label for the hyperlink for alternate URL. If not set, the hyperlink label will default to "Open in New Window" for a URL Portlet and "Show Details" for a Report Portlet. Refer to "Internationalize the Dashboard Page" for details on how to internationalize ALT_URL_LABEL.

Table 9–2 RSS Portlet Input Parameters

Parameter	Description
RSS_URL	This is the URL of the RSS feed that the portlet will display in its content space.
PORTLET_WIDTH	This is an optional parameter. The RSSPortlet width defaults to 325. If this parameter is set, it allows the dashboard developer to override the default width of the portlet.
PORTLET_HEIGHT	This is an optional parameter. The RSSPortlet height defaults to 250. If this parameter is set, it allows the dashboard developer to override the default height of the portlet.
SHOW_ITEM_DESCRIPTION	<p>The RSSPortlet displays a list of the RSS feed's <item> elements. Each <item> element is represented as a single hyperlink whose text is the value of the item's <title> element. The portlet displays one link per line.</p> <p>Some RSS Feeds have a description associated with items. By default, RSSPortlet will not display the contents of the description below the item. If this optional parameter is set to TRUE, the portlet will display the contents of the description below the item.</p>

Table 9–3 ORWOBIEEAlerts Portlet Input Parameters

Parameter	Description
ORW_OBIEE_ALERTS_APPLICATION_URL	This is the URL of the OBIEEAlerts Application deployment that the portlet will display in its content space.
OBIEE_MANAGE_ALERTS_URL	This is an optional parameter. If specified, the portlet will have a Hyperlink enabled on the top - right that will launch the passed Oracle BI Delivers URL in a separate window
OBIEE_CONTEXT_ROOT	This is an optional parameter. This parameter specifies the Context Root for the Oracle BI EE Delivers deployment. It defaults to <code>analytics</code> . If your Oracle BI EE Delivers deployment uses a different context root you must set this parameter, as it is passed to the ORWOBIEEAlerts application, which uses it to construct the Oracle BI EE RSS Feed URL.
PORTLET_WIDTH	This is an optional parameter. The ORWOBIEEAlertsPortlet width defaults to 325. If this parameter is set, it allows the dashboard developer to override the default width of the portlet.
PORTLET_HEIGHT	This is an optional parameter. The ORWOBIEEAlertsPortlet height defaults to 250. If this parameter is set, it allows the dashboard developer to override the default height of the portlet.

Table 9–4 ORW Slideshow Portlet Input Parameters

Parameter	Description
PORTLET_WIDTH	This is an optional parameter. The ORWSlideshowPortlet width defaults to 400. If this parameter is set, it allows the dashboard developer to override the default width of the portlet.
PORTLET_HEIGHT	This is an optional parameter. The ORWSlideshowPortlet height defaults to 300. If this parameter is set, it allows the dashboard developer to override the default height of the portlet.

Table 9–4 (Cont.) ORW Slideshow Portlet Input Parameters

Parameter	Description
PORTLET_TITLE	This is an optional parameter. The ORWSlideshowPortlet title defaults to ORW Slideshow Portlet. If this parameter is set, it allows the dashboard developer to override the default title of the portlet.
SLIDESHOW_ON	This is an optional parameter. The ORW SlideshowPortlet defaults to 'false'. If this parameter is set, it allows the dashboard developer to override the default slideshow on value for the portlet. If this value is set to 'true', the ORWSlideshow will start in play mode. Legal values are 'true' and 'false'.
SLIDE_SPEED	This is an optional parameter. This parameter sets the ORWSlideshowPortlet slide speed in seconds. The slide speed is the amount of time a given slide is displayed until it transitions to the next configured slide while in play mode. If this parameter is not set, it defaults to 20 seconds. If this parameter is set, it allows the dashboard developer to override the default slide speed of the slideshow portlet. Valid values are any whole positive number between 1 and 999.
TRANSITION_TIME	This is an optional parameter. This parameter sets the ORWSlideshowPortlet transition time in milliseconds. The transition time is the amount of time spent on the transition effect when going from one slide to the next. If this parameter is set, it allows the dashboard developer to override the default transition time of the slideshow portlet. Ensure that the transition_time is a positive whole number. In addition, it is recommended that the transition_time be less than the slide_speed by a minimum of 250 milliseconds.
TRANSITION_EFFECT	This is an optional parameter. This parameter sets the ORWSlideshowPortlet transition effect. The valid transition effects are—wipe, cover, uncover, turnUp, and turnDown. When this parameter is not set, it defaults to wipe. If this parameter is set, it allows the dashboard developer to override the default transition effect of the slideshow portlet.
OPEN_IN_NEW_WINDOW_LINK_RENDERED	This is an optional parameter. This parameter determines whether or not the 'Open in New Window' link is displayed for the ORWSlideshowPortlet. When not configured, the default value is 'false'. If this parameter is set, it allows the dashboard developer to specify whether the Open in New Window link should be rendered. Valid values are 'true' and 'false'.
OPEN_IN_NEW_WINDOW_LINK_TEXT	This is an optional parameter. This parameter displays the 'Open in New Window' link text for the ORWSlideshowPortlet. When not configured, the default value is 'Open in New Window'. If this parameter is set, it allows the dashboard developer to override the default Open in New Window link text of the slideshow portlet. If the OPEN_IN_NEW_WINDOW_LINK_RENDERED value is false, then this parameter will be ignored.

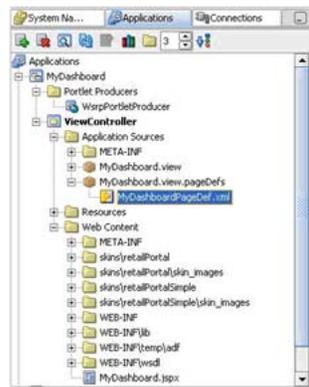
Table 9–4 (Cont.) ORW Slideshow Portlet Input Parameters

Parameter	Description
SLIDE1_URL	Sets the URL for each of the eight slide URL parameters. Each configured URL must be able to render correctly within an HTML <iframe>.
SLIDE2_URL	
SLIDE3_URL	
SLIDE4_URL	
SLIDE5_URL	
SLIDE5_URL	
SLIDE7_URL	
SLIDE8_URL	
	If a slide URL parameter is not set, there is no default value and the slide will not be displayed.
	If only one slide is configured, the portlet will display the single URL without the slideshow navigation controls. This will be similar to the URL portlet.
	If two or more URLs are configured, the portlet will display the slideshow navigation controls, and the configured URLs will operate as a slideshow.
	The slide URLs do not need to be configured in order. It is valid to enter a URL for Slide1, skip Slide2, and enter a URL in Slide3. Any slide URLs that are not configured will be ignored.
	If no URLs are configured, an information message will appear stating that there is no content to display.
SLIDE1_ALT_URL	Sets the alternate URL for the corresponding slide, which is the URL to display when the 'Open in new Window Link' is clicked.
SLIDE2_ALT_URL	
SLIDE3_ALT_URL	
SLIDE4_ALT_URL	
SLIDE5_ALT_URL	
SLIDE6_ALT_URL	
SLIDE7_ALT_URL	
SLIDE8_ALT_URL	
	This value will be ignored if the OPEN_IN_NEW_WINDOW_LINK_RENDERED parameter is set to false or if the corresponding SLIDE#_URL parameter is not set.
	If the OPEN_IN_NEW_WINDOW_LINK_RENDERED parameter is set to true, and the SLIDE#_ALT_URL is not configured, the 'Open in New Window' link will launch the SLIDE#_URL in the new window.
SLIDE1_TITLE	Sets the corresponding slide title. If the slide title is not configured and a slide URL is configured, the value of the slide title will default to 'Slide <n>' where <n> is the corresponding slide number (1 through 8).
SLIDE2_TITLE	
SLIDE3_TITLE	
SLIDE4_TITLE	
SLIDE5_TITLE	
SLIDE6_TITLE	
SLIDE7_TITLE	
SLIDE8_TITLE	
	The slide title will appear in the navigation drop-down of the slideshow portlet.
	If the corresponding slide URL is not configured, then this value will be ignored.

More Information about Page Definition Files

Page definition files define the binding objects that populate the data in UI components at runtime. They are located under `ViewController\Application Sources`. The Application Navigator showing the location of `MyDashboardPageDef.xml` is shown in the following figure.

Figure 9–7 Application Navigator Showing the Location of MyDashboardPageDef.xml

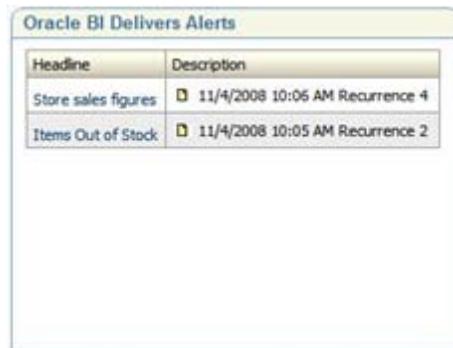


For more information, refer to the Page Definition section of the *ADF Developer's Guide*.

OBIEEAlerts Portlet Example

OBIEE Alerts Portlet, OBIEE_MANAGE_ALERTS_URL parameter not specified

Figure 9–8 Example of OBIEEAlerts Portlet



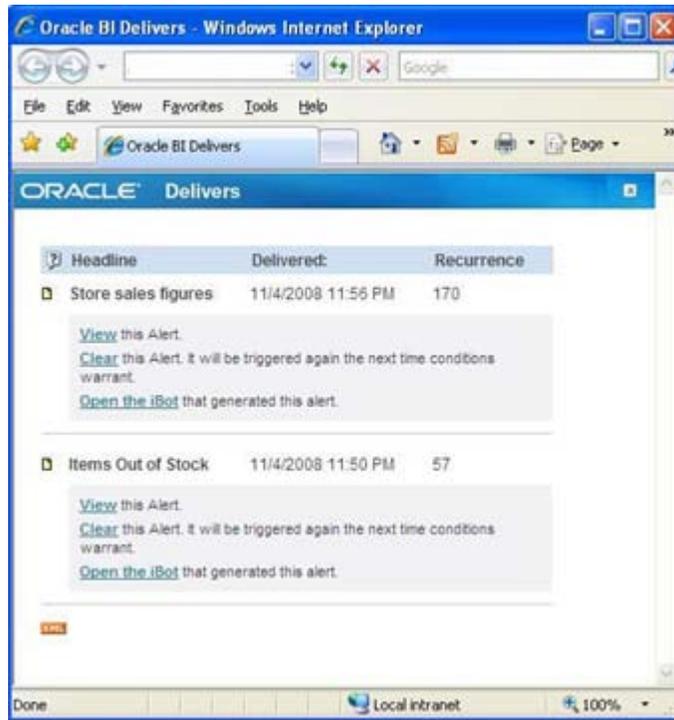
OBIEE Alerts Portlet, OBIEE_MANAGE_ALERTS_URL parameter specified

Figure 9–9 OBIEEAlerts Portlet with the Manage Alerts Parameter Specified



Clicking on the Manage Alerts link will take you to the following BI Delivers Screen that allows you to manage your alerts.

Figure 9–10 Oracle BI Delivers Screen



URL Portlet Example

Figure 9–11 Example of an URL Portlet



Report Portlet Example

Figure 9–12 Example of a Report Portlet



RSS Portlet Example

RSSPortlet with SHOW_ITEM_DESCRIPTION=FALSE

Figure 9-13 Example of an RSS Portlet



RSSPortlet with SHOW_ITEM_DESCRIPTION=TRUE

Figure 9-14 Example of an RSS Portlet with SHOW_ITEM_DESCRIPTION=TRUE



ORW Slideshow Portlet Examples

ORW Slideshow Portlet with 2 or more configured URLs

Figure 9-15 Example of an ORW Slideshow Portlet



ORW Slideshow Portlet with 1 configured URL

Figure 9–16 Example of the ORW Slideshow Portlet With a Configured URL



ORW Slideshow Portlet with no configured URLs

Figure 9–17 Example of the ORW Slideshow Portlet With No Configured URLs



Adding Oracle Retail Skins to your Dashboard

In this lesson you will learn about ADF Faces skins and how to add the Oracle Retail skins to your dashboard.

What is a Skin?

An Oracle ADF Faces skin applies a global style to the entire application. You can use style selectors in your own custom skin to modify selected aspects of a component or area of a component.

Add the Oracle Retail Skin to your Dashboard

By default, JDeveloper configures ADF Faces applications to use the Oracle skin. The ORW application has been packaged with a custom skin. To be consistent with the default ORW look and feel, the same skin needs to be added to dashboards that are developed for ORW.

Note: If you have customized the ORW skins, you should use your customized skins rather than the Oracle Retail skins.

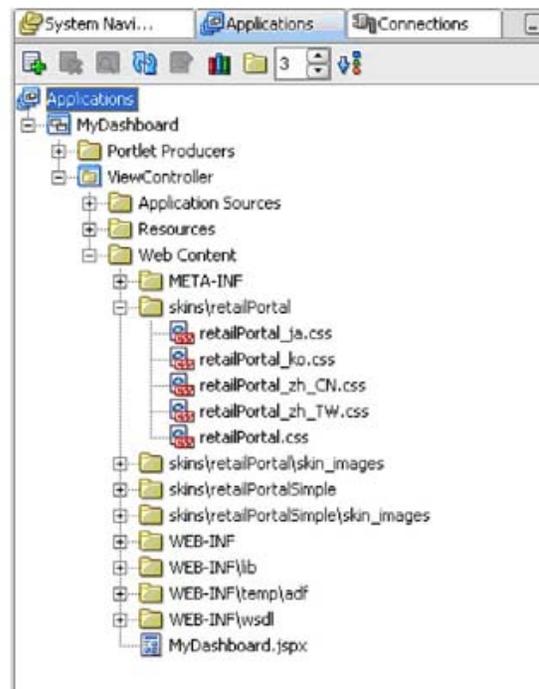
To add Oracle Retail skins:

1. The RetailWorkspaceApplication.zip file contains a dashboard development pack zip file - orw-dashboards-devpack.zip. The Retail Workspace skin jar is included in the dashboard development pack zip file. Extract this `oretail-skins.jar` file from `orw-dashboards-devpack.zip`, and then unzip its contents to the `ViewController\public_html` directory of your MyDashboard application.

Note: The skins jar file includes an `adf-faces-skins.xml` file. This file contains the definition of the skin family provided by the Workspace. If you would like to preserve your own `adf-faces-skins.xml`, back it up first and then merge the two files.

2. In the JDeveloper Applications Navigator, refresh the MyDashboard ViewController Project to reflect the changes to the MyDashboard application. You should now see `skins\retailPortal` and `skins\retailPortalSimple` folders and sub-folders below `ViewController\Web Content`. You will be using the skins from `skins\retailPortal` as shown in the following figure.

Figure 9–18 Applications Navigator



3. Extract the `oretail-common.jar` file from `orw-dashboards-devpack.zip` and copy the intact jar into the `ViewController\public_html\WEB-INF\lib` directory of the MyDashboard application. Do not unzip its contents.

The `oretail-common.jar` contains the `SkinsBean` that is responsible for retrieving the appropriate Oracle Retail skin based on locale.

4. In the Application Navigator, expand the `ViewController/Web Content/WEB-INF` folder. Open `faces-config.xml` and edit the configuration file to include the `SkinsBean` managed bean, as shown in the following example:

```
<managed-bean>
  <managed-bean-name>skins</managed-bean-name>

<managed-bean-class>oracle.retail.common.faces.bean.SkinsBean</managed-bean-class>
</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>baseSkinFamily</property-name>
    <property-class>java.lang.String</property-class>
    <value>retailPortal</value>
  </managed-property>
  <managed-property>
    <property-name>baseStyleSheetName</property-name>
    <property-class>java.lang.String</property-class>
    <value>retailPortal</value>
  </managed-property>
  <managed-property>
    <property-name>skinsDirectory</property-name>
    <property-class>java.lang.String</property-class>
    <value>skins/retailPortal</value>
  </managed-property>
</managed-bean>
```

For more information about the `SkinsBean`, refer to [Chapter 8, "Customization Guide"](#).

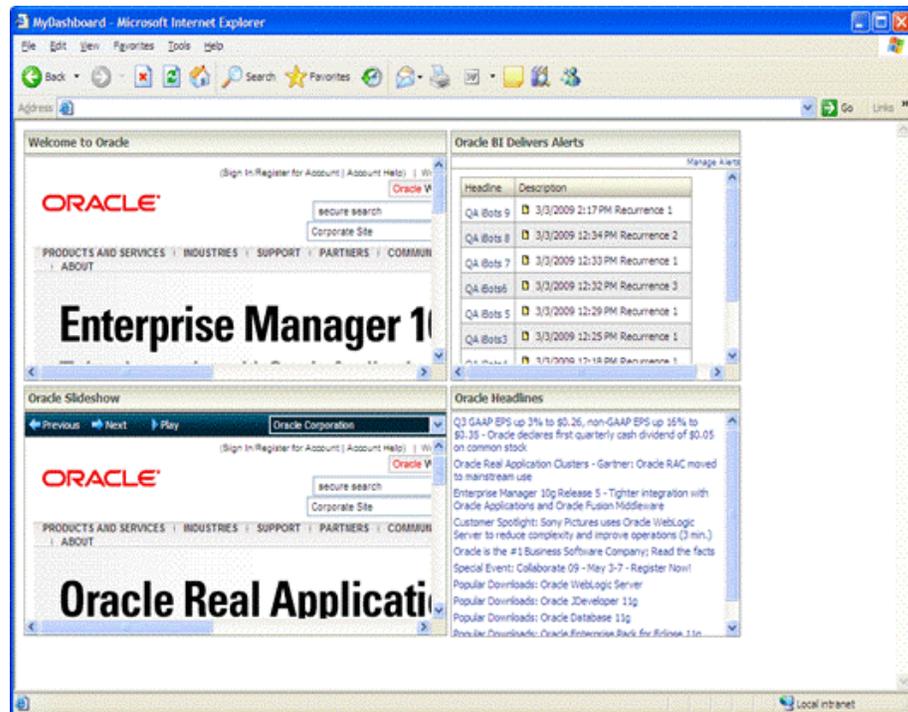
5. Configure the MyDashboard application to use the skin.
 - a. In the Application Navigator, expand the `ViewController/Web Content/WEB-INF` folder. Open `adf-faces-config.xml` and edit the configuration file.
 - b. In `adf-faces-config.xml`, the `<skin-family>` element configures the skin to be used. Replace the default value for `<skin-family>` (usually `oracle`) with an EL expression to get the skin name from the `SkinsBean`. The element should look like the following snippet of XML:

```
<skin-family>#{skins.skinFamily}</skin-family>
```

6. Right-click on the `MyDashboard.jspx` page in the Application Navigation pane and click **Run** in the context menu to test the dashboard with the new skin.

The following figure depicts the after effects of applying the skins. The most noticeable changes in appearance are the change in the font used in the RSS feed, and the change in the portlet header font and background.

Figure 9–19 Dashboard After Applying a Skin



Troubleshooting

If you have installed the ORW skins but do not see the expected changes in appearance, do the following:

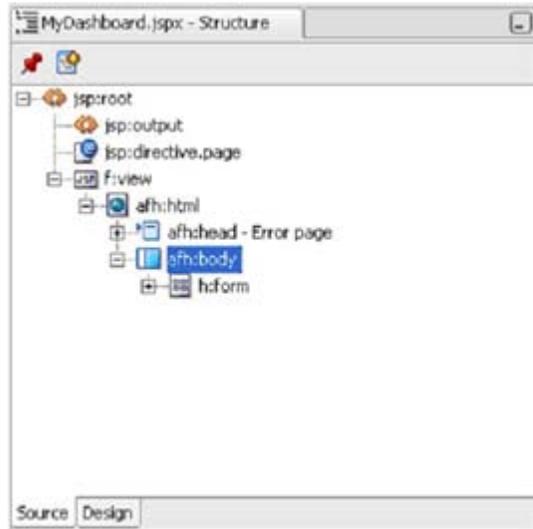
- Click the browser **Refresh** button to refresh your page. ADF Faces caches skin settings. Sometimes it is necessary to refresh the page to clear the cache.
- If refreshing the page does not work, open the Internet Explorer Internet Options dialog and delete all temporary internet files.
- Make sure you copied the skins directory, including all skin files and subdirectories from `oretail-skins.jar` to below `ViewController\public_html`.
- Make sure you copied `oretail-common.jar` to the `WEB-INF\lib` directory of your MyDashboard project workspace.
- Make sure your `adf-faces-skins.xml` includes the `<skin>` elements that define the ORW skins. A sample version of this file is included in `oretail-skins.jar`.
- Make sure your `faces-config.xml` has been modified to configure a **skins** managed bean that references the `SkinsBean`.
- Make sure your `adf-faces-config.xml` has been modified to specify `#{skins.skinFamily}` as the value of the `<skin-family>` element.

Adding the `retailPortalContentBody` Style Class to Dashboard Body

When you display the dashboard in ORW, there is some extra space around the left and top of the dashboard as it gets rendered within the content area. To minimize this, you can use a style class that is defined in the `retailPortal` skin.

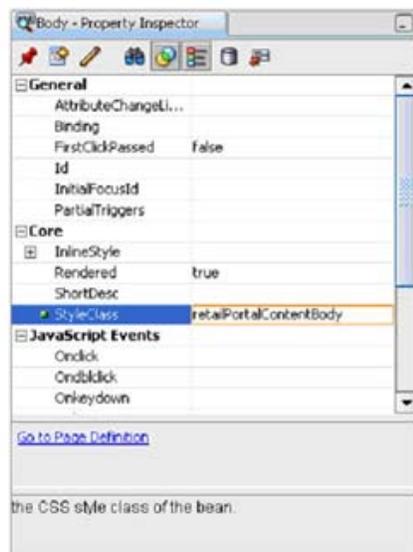
1. In the Applications Navigator, expand the View-Controller node and then expand the Web Content node. Select MyDashboard.jspx and open the page in the editor pane.
2. In the MyDashboard.jspx Structure pane, expand the jsp:root, f:view and afh:html nodes and select the afh:body.

Figure 9–20 Structure Pane



3. In the Property Inspector for the Body component, set the value of the StyleClass property to retailPortalContentBody. This style class is defined in the Retail Workspace skin. The style class sets the default padding and insets of the body to 0, which removes any default padding and insets.

Figure 9–21 Property Inspector



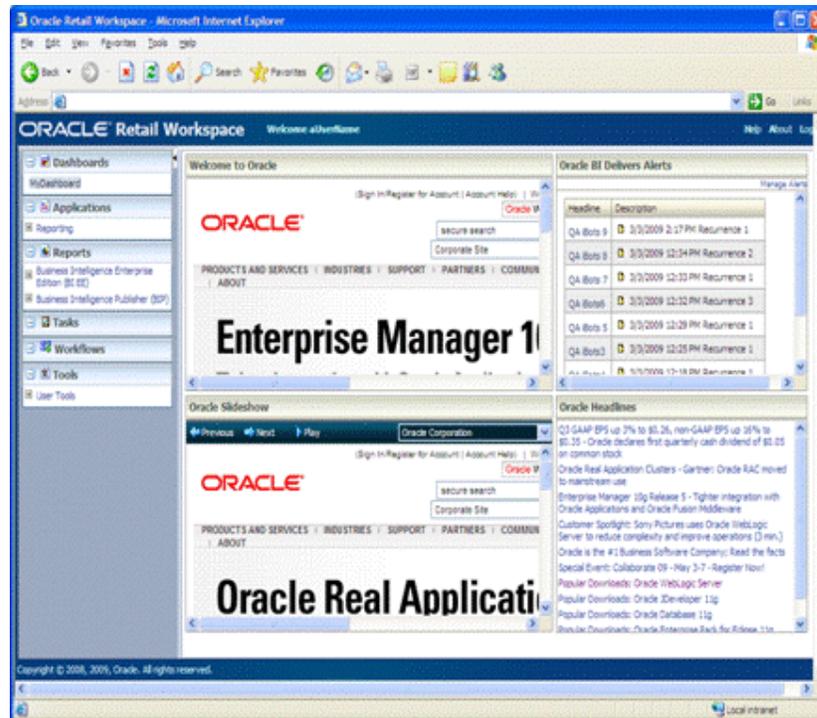
4. Now that you have removed the default padding and insets, you may find that you need to add backspace around the portlets. You may need to insert

ObjectSpacer components as necessary to the left of the first portlet in each row and to the right of the last portlet in each row. Set the width property of the ObjectSpacer to 4 to match the horizontal spacing you set in a previous chapter.

You may also have to insert an ObjectSpacer before the first inner PanelCustomizable and after the last inner PanelCustomizable. Set the height property to 4 to match the vertical spacing you set in a previous chapter.

The following figure shows MyDashboard after making these changes in Workspace.

Figure 9–22 Dashboard After Making the Changes



Securing the Dashboard

This lesson explains how to secure your dashboard so that only authorized users can access it. Security for ADF applications is explained further in Chapter 18 of the *Oracle® Application Development Framework Developer's Guide*.

So far in this tutorial, the dashboard has no security mechanisms in place. It is simply a JSF application that may be deployed and is open to anybody. This section details how to add security mechanisms to the dashboard to insure only authorized users are allowed to access the dashboard.

There are multiple security mechanisms in place with ADF and the ORW implementation. The first involves standard J2EE Web application security. Standard container managed security involves the Java Authentication and Authorization Service (JAAS). Oracle's implementation of this service is known as JAZN.

Two basic concepts of security are:

- Authentication—ensuring a user is who the user claims to be.
- Authorization—allowing (or denying) a user specific capabilities.

For a dashboard, authentication is provided by the application server container which may delegate this to another subsystem, such as Oracle Single Sign-On (OSSO). This allows a user to enter credentials (user ID and password) once per HTTP session. In order to leverage the OSSO subsystem, the dashboard must be configured with specific entries in its deployment descriptors.

The first deployment descriptor is the `web.xml` file. This is a standard J2EE file included with all web applications. There are a few requirements for this file:

- It must define the ADF Authentication servlet, including initialization parameters.
- It must define the URI of the ADF Authentication servlet.
- It must define an authentication security constraint on the ADF authentication servlet.
- It must declare all logical roles referenced by the ADF Authentication servlet.

Authorization for accessing specific JSPX pages is provided by the ADF infrastructure as well. ADF provides mechanisms to control fine-grained access to JSPX files and to resources found on these files, such as database connections. When correctly configured, the ADF infrastructure will deny access without an application developer writing any security specific code. However, there are specific configuration entries required for this.

The final source of security is supplied by the ORW framework. This software verifies that a user only sees the work lists, external links, and dashboards appropriate for a user.

In summary, the following needs to be performed to ensure a dashboard is secure:

1. Run the ADF Security wizard.
2. Edit the `web.xml` deployment descriptor.
3. Create the `orion-web.xml` deployment descriptor file to correctly map the logical role for a dashboard to an actual or physical role.
4. Update the `adf-config.xml` file to ensure page-level security is in place.
5. Edit the `system-jazn-data.xml` file for the embedded OC4J for use with all of the above. Add a permission grant to allow access to the dashboard.

Note: The instructions above use the OC4J XML file security provider to add security. Details on how to convert from the XML file security provider to the OID LDAP security provider are explained later.

Run the ADF Security Wizard.

To run the ADF security wizard:

1. Highlight **ViewController** in the Navigation pane and select the **ADF Security Wizard** under the Tools menu.
2. If a Welcome window appears, click **Next**. This should bring up the following display:

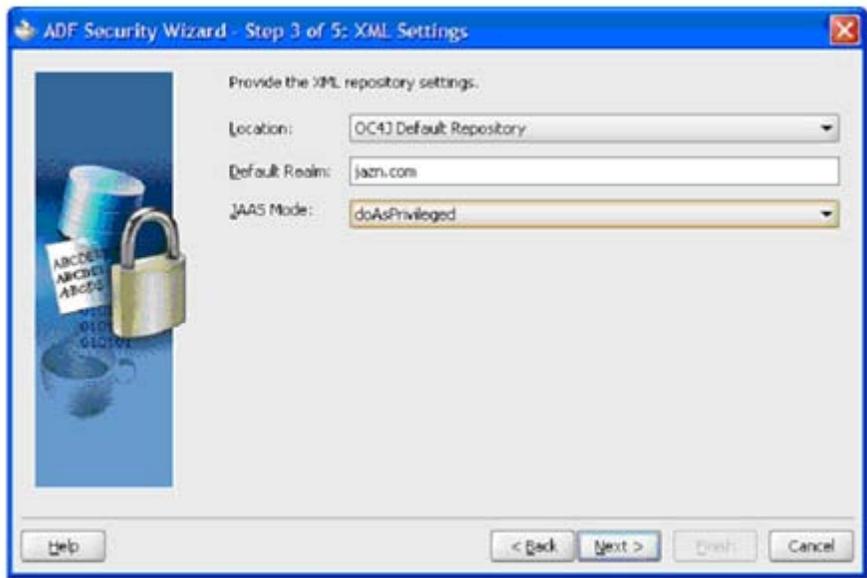
Figure 9–23 ADF Security Wizard - Authentication Screen

3. Check the **Enforce Authorization** checkbox and click **Next**.

Figure 9–24 ADF Security Wizard - Oracle JAAS Provider Screen

4. Check **Lightweight XML Provider** and click **Next**.

Figure 9–25 ADF Security Wizard - XML Settings Screen



5. Specify the OC4J Default Repository, the jazzn.com default repository, and a JAAS Mode of `doAsPrivileged`. Click **Next**.

Figure 9–26 ADF Security Wizard - Login Screen



6. Check **HTTP Basic Authentication**. Leave the Realm empty and click **Next**.

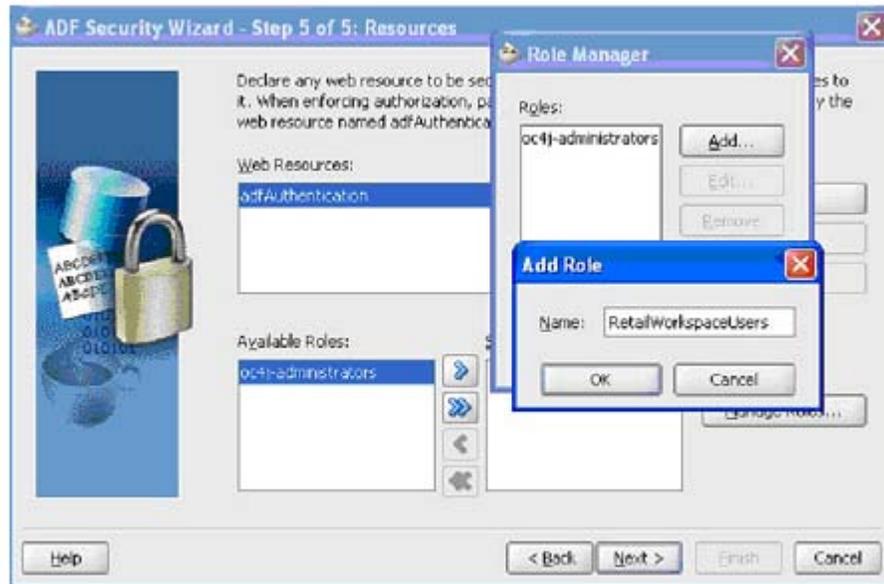
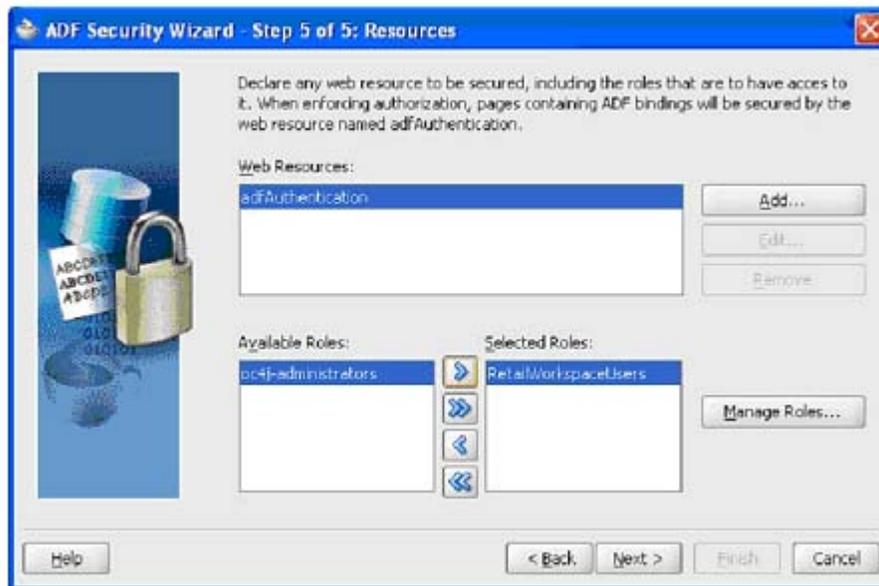
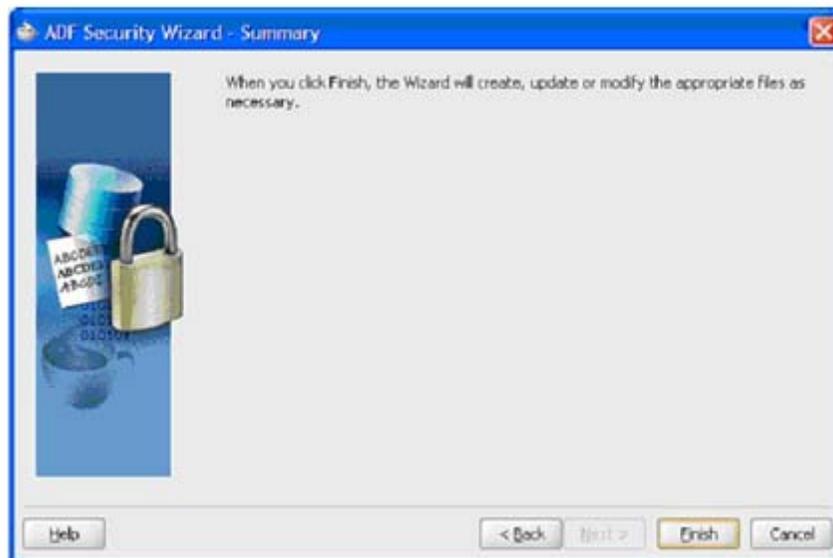
Figure 9–27 ADF Security Wizard - Resources Screen**Figure 9–28 Add Role Window**

Figure 9–29 Resources Screen With the New Role Selected



7. On this screen, you may see **oc4j_administrators** as a Selected Role. Move that to the Available roles and click on **Manage Roles** to define the logical roles for the application. This brings up the Role Manager dialog. Add **RetailWorkspaceUsers** as a new role and click **OK**. Make sure **RetailWorkspaceUsers** is in the Selected Roles box and click **Next** to bring up the Summary page:

Figure 9–30 ADF Security Wizard - Summary Screen



8. To close the wizard, click **Finish**.

Edit the web.xml Deployment Descriptor

The following steps create the logical role, `RetailWorkspaceUsers`, and add a security constraint that forces a user to be an authenticated member of this group in order to access your page. ADF security will also enforce that the user has been granted the correct permissions.

1. Open the `ViewController` folder. Select **Web Content**, then **public_html**, then **WEB-INF**, and double-click on the `web.xml` file. There are three sections of text that need to be placed within the `<web-app>` tag in this file:
2. The `adfAuthentication` servlet must be defined in `web.xml`. ADF applications delegate authentication and authorization activities to the `adfAuthentication` servlet. This servlet must be defined in the `web.xml` file or else the application server will not allow any requests made to it. The entries that define the ADF authentication servlet are shown in the following example:

```
<servlet>
  <servlet-name>adfAuthentication</servlet-name>
  <servlet-class>
oracle.adf.share.security.authentication.AuthenticationServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
  <security-role-ref>
    <role-name>RetailWorkspaceUsers</role-name>
    <role-link>RetailWorkspaceUsers</role-link>
  </security-role-ref>
</servlet>

<servlet-mapping>
  <servlet-name>adfAuthentication</servlet-name>
  <url-pattern>/adfAuthentication/*</url-pattern>
</servlet-mapping>
```

The `<servlet>` tag defines the servlet and the fact that the logical role name, `RetailWorkSpaceUsers`, is referenced by servlet. The `<servlet-mapping>` tag defines the URI used to access the ADF Authentication servlet.

3. Apply authentication constraints to the ADF authentication servlet. An authentication constraint specifies to the application server that only authenticated users with a specific logical role are allowed to access a set of URLs. See the following example:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>adfAuthentication</web-resource-name>
    <url-pattern>/adfAuthentication</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>RetailWorkspaceUsers</role-name>
  </auth-constraint>
</security-constraint>
```

Note: Applying an authentication constraint to the ADF Authentication servlet allows the non-authenticated users to become authenticated via interactions with the application. Alternatively, one can create a authentication constraint on the Faces Servlet which will force all the users to become authenticated before entering the application.

4. Declare the logical roles used by the Dashboard and the authentication constraints. Proper `web.xml` files declare all logical roles. See the following example:

```
<security-role>
  <role-name>RetailWorkspaceUsers</role-name>
</security-role>
```

5. Finally, you need to add a mechanism for the user to authenticate (log in) with. This is done via the `<login>` element. The text below specifies BASIC authentication, in which a standard pop-up window is used for logging in.

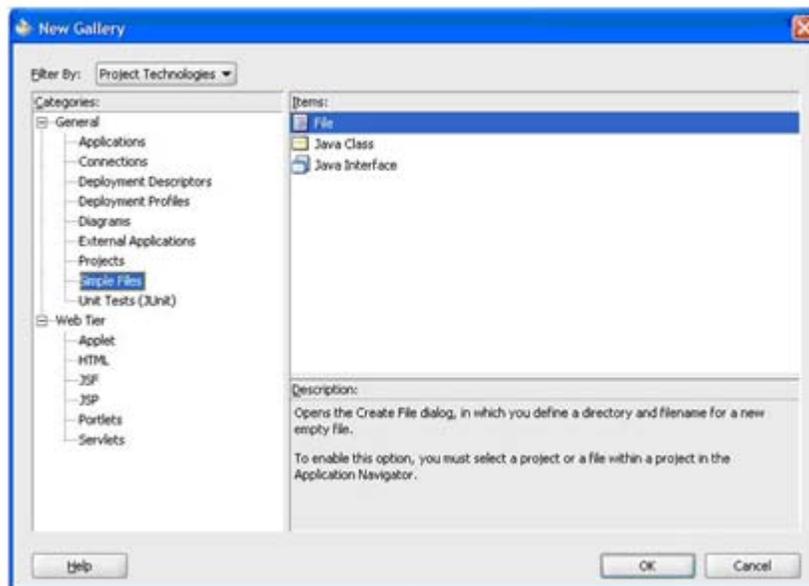
```
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
```

Create the `orion-web.xml` Deployment Descriptor

You need to create the `orion-web.xml` deployment descriptor to correctly map a dashboard's logical role to an actual role.

1. Right-click on the Web Content, then select **public_html**, and then the **WEB-INF** folder. Select the **New** menu item.
2. From the New Gallery window, under Categories select **Simple Files** and under Items select **File** as shown below.

Figure 9–31 New Gallery Window



3. Click **OK** and create the file named `orion-web.xml` in the **WEB-INF** directory:

Figure 9–32 Create File Window

- Put the following text into the new `orion-web.xml` file:

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<orion-web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://xmlns.oracle.com/oracleas/schema/orion-web-10_0.xsd"
  schema-major-version="10" schema-minor-version="0"
  servlet-webdir="/servlet/">
  <security-role-mapping name="RetailWorkspaceUsers" impliesAll="false">
    <group name="Retail_Workspace_Users"></group>
  </security-role-mapping>
  <jazn-web-app runas-mode="true" doasprivileged-mode="true"/>
</orion-web-app>
```

Create/Update the `adf-config.xml` File

You need to create or update the `adf-config.xml` file to ensure page-level security is in place.

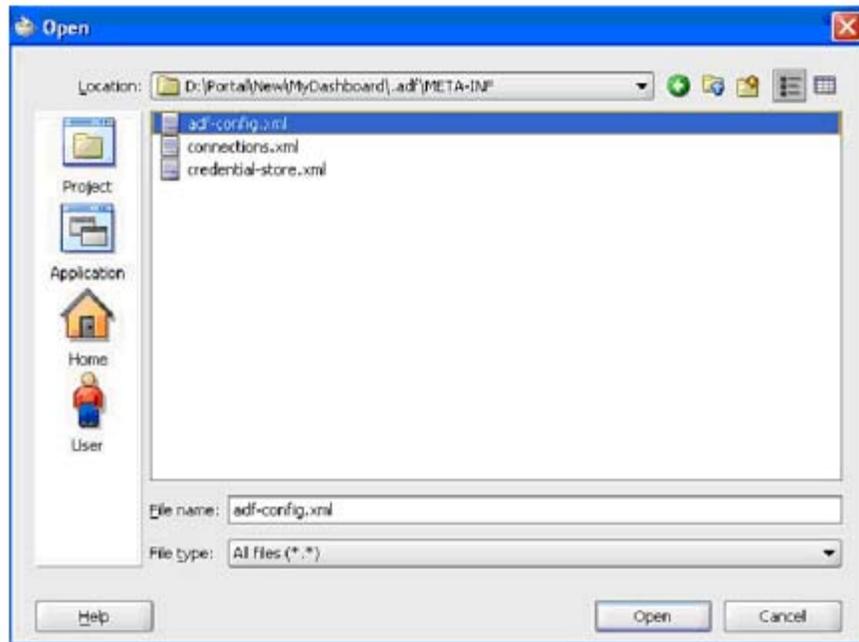
If the `adf-config.xml` file does not exist, you need to create a new one.

To determine if the file exists, click on File and then open. Try to open the following file:

```
<Dashboard application dir>/adf/META-INF/adf-config.xml
```

where `<Dashboard application dir>` is the root directory of the dashboard application. Note that the `.adf` folder is found in the same folder on the file system as the `ViewController` folder.

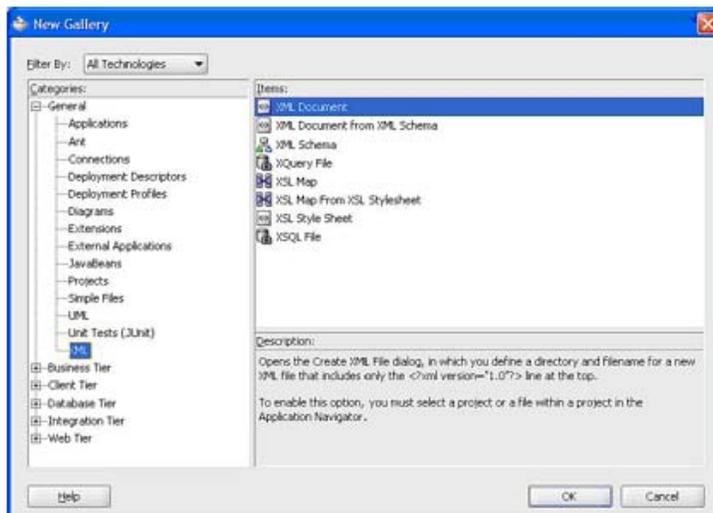
Figure 9–33 Open Dialog Box



If the `adf-config.xml` file already exists, open it and skip the following section.

1. Cancel out of the Open dialog.
2. Right-click on the View-Controller project and select the **New** menu item.
3. From the New Gallery window, find the **Filter by** menu and select **All Technologies**.
4. Under Categories, select **XML**.
5. Under Items, select **XML Document** as shown below.

Figure 9–34 New Gallery Window



6. To bring up the following dialog window, click **OK**:

Figure 9–35 Create XML File Dialog Box

7. Enter the File Name of `adf-config.xml` and a directory name of `<application directory>/ .adf/META-INF` (`<application directory>` is where the JDeveloper application files are stored). To search for a directory, click **Browse**.
8. Once `adf-config.xml` exists, open the file via the File open dialog. Then, modify the file as detailed in the section below.

Skip to here if the `adf-config.xml` file already exists.

At the bottom of this file just before the closing `</adf-config>` tag, you will see the `<adf-config-child>` element. This element should contain an empty `<JaasSecurityContext/>` element. Replace this element with the one detailed below:

```
<JaasSecurityContext initialContextFactoryClass="oracle.adf.share.security.
JAASInitialContextFactory" authorizationEnforce="true" jaasProviderClass=
"oracle.adf.share.security.providers.jazn.JAZNSecurityContext" >
</JaasSecurityContext>
```

The `<adf-config-child>` element will most likely contain another element besides the `<JaasSecurityContext>`. This additional element is the `<CredentialStoreContext>` element and is used to store user credentials used with portlet producers. If this tag exists, you should keep it untouched. In this scenario, the end of the `adf-config` file would look similar to the following:

```
<adf-config-child xmlns="http://xmlns.oracle.com/adf/security/config">
  <CredentialStoreContext credentialStoreClass="oracle.adf.share.security.
providers.jazn.JAZNCredentialStore"
    credentialStoreDefaultUser="anonymous"
    credentialStoreLocation="./credential-store.xml" />
  <JaasSecurityContextinitialContextFactoryClass="oracle.adf.share.
security.JAASInitialContextFactory"authorizationEnforce="true"jaasProviderClass
="oracle.adf.share.security.providers.jazn.JAZNSecurityContext" >
    </JaasSecurityContext>
</adf-config-child>
</adf-config>
```

After modifying the `adf-config.xml` file, save and close the file.

Edit the system-jazn-data.xml File and Add a Permission Grant

Note: The following instructions edit the embedded OC4J's `system-jazn-data.xml` file. Alternatively, you can use the Tools->Embedded OC4J System Preferences to achieve the same results. See the *ADF Developers Guide* for more details.

Edit the `system-jazn-data.xml` file for the embedded OC4J and add a permission grant to allow access to the dashboard.

All of the previous steps allow the dashboard application to be securely deployed. But the embedded OC4J (the application server where you test the dashboard) must have the correct security configuration to allow access to the dashboard. This configuration includes roles, users, and permission grants to these roles and users.

JDeveloper deploys applications to the embedded OC4J configured to use the XML File-based Security provider. This means that all of the security information is provided in the embedded OC4J `system-jazn-data.xml` file. The location of this file is the folder:

```
<JDeveloper Install>/jdev/system/oracle.j2ee.10.1.3.<XX.YY>/
embedded-oc4j/config
```

<JDeveloper Install> is the directory where JDeveloper is installed. <XX.YY> is the system release version, for example, 41.57.

The first thing to do to `system-jazn-data.xml` is to create a user. To create a new user ID in this file, search for the <users> tag and add in a <user> element to define the new user's name, display name, and password.

```
<user>
  <name>aUserName</name>
  <display-name>A User Name</display-name>
  <description>A description </description>
  <credentials>!aPassword</credentials>
</user>
```

where:

- *aUserName* should be the desired user name ID
- *A User Name* should be the display name of the user ID
- *A description* should be a short description of the user ID
- *aPassword* is the user's password. Make sure that this entry begins with an exclamation mark. The password used should not contain an exclamation mark.

If you were to add in the example <user> element verbatim, you would log in with the user name of 'aUserName' and a password of 'aPassword'.

The embedded OC4J reads this entry the next time it is started. At that time, it replaces the string '!aPassword' with a one-way hash value. It also generates a unique GUID string.

Next, you need to create the role for the user. Search the `system-jazn-data.xml` file for the <roles> element and create two roles: `Retail_Workspace_Users` and the roles that will be granted permission to access the dashboard. For example,

```
<role>
  <name>A_Role_Name</name>
```

```

<members>
  <member>
    <type>user</type>
    <name>aUserName</name>
  </member>
</members>
</role>
<role>
  <name>Retail_Workspace_Users</name>
  <members>
    <member>
      <type>role</type>
      <name>A_Role_Name</name>
    </member>
  </members>
</role>

```

where:

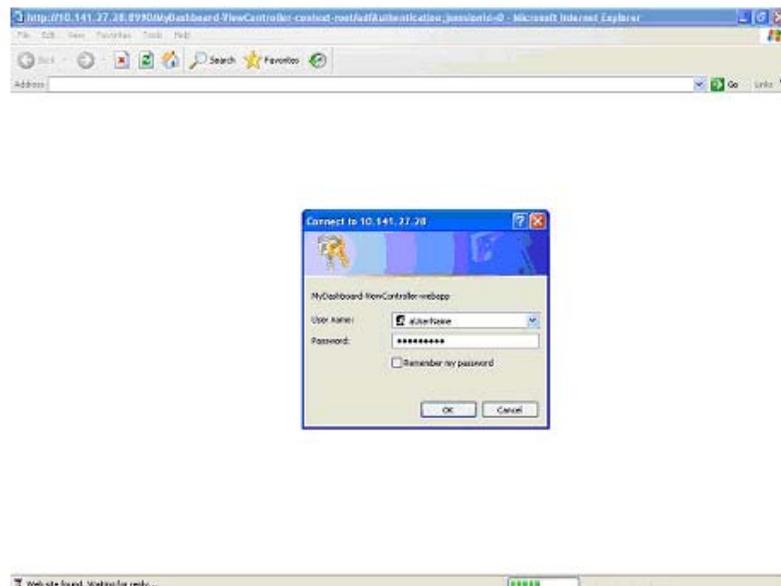
- *aUserName* should be the desired user name ID mentioned above
- *A_Role_Name* should be the new role name

Note: The definition of the **A_Role_Name** role must come before the definition of the **Retail_Workspace_Users** role.

You can add multiple users and roles to an existing role.

Once you have edited the `system-jazn-data.xml` file of the embedded OC4J, you should be able to run your dashboard and be prompted for a user name and password.

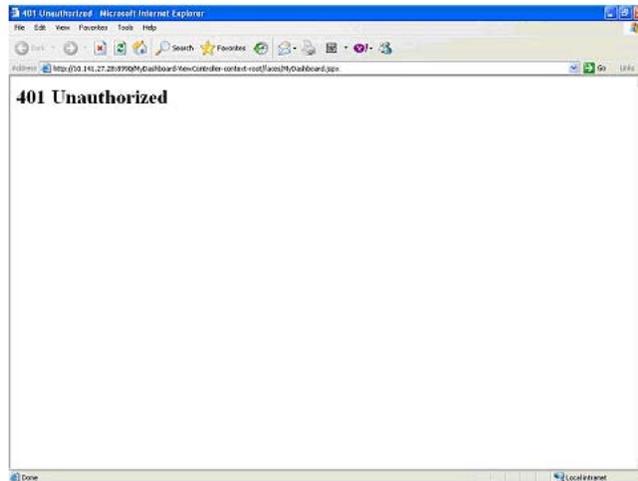
Figure 9-36 Login Prompt to Access the Dashboard



Once logged in, though, you will not be able to access your dashboard. You will get a 401 Unauthorized error. The reason is that you now have a secured application, but

you have not granted permission to any users to access it. In effect, you have an application that nobody is allowed to use.

Figure 9–37 401 Unauthorized Page



To grant permissions, you must edit the `system-jazn-data.xml` file and add a grant to the JAZN policy. The JAZN policy is encapsulated in the `<jazn-policy>` element. ADF requires a specific type of entry here to allow access to a given page. To grant access to view a JSPX page, you create a permission grant for the page definition file for that page.

See the following example:

```
<grant>
<grantee>
  <principals>
    <principal>
      <type>role</type>
      <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
      <name>jazn.com/A_Role_Name</name>
    </principal>
  </principals>
</grantee>
<permissions>
  <permission>
    <class>oracle.adf.share.security.authorization.RegionPermission</class>
    <name>pageDefinitionpackage.pageDef_ID</name>
    <actions>view</actions>
  </permission>
</permissions>
</grant>
```

where:

- **A_Role_Name** is the name of the role to be granted permission.
- **pageDefinitionpackage.pageDef_ID** is the JSPX page definition file package name and ID. To get the correct value, right-click on the jsp page in the Application Navigation pane and click **Go to Page Definition**. The package name and ID is found within the `<pageDefinition>` element. For example,

```
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
```

```
version="10.1.3.41.57" id="MyDashboardPageDef"
Package="MyCompany.view.pageDefs">
```

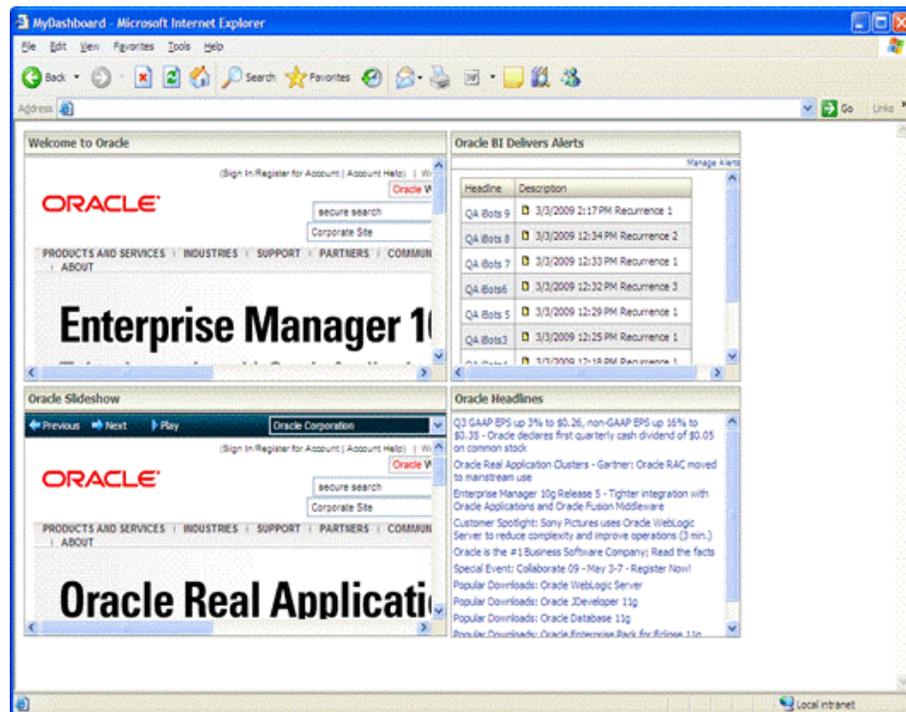
results in the string:

```
MyCompany.view.pageDefs.MyDashboardPageDef
```

Note: If the same role is to access multiple pages, you may put multiple <permission> tags into a single <permissions> element. However, this same technique does not apply to the <principal> element. Do not place multiple <principal> tags within a single <principals> element, as the semantics are not the same and cannot be managed within Workspace.

As with new users and roles, the embedded OC4J must be restarted before it can see any new permission grants. At this point, the new dashboard has been made secure, and only authorized users can access the page.

Figure 9–38 A Secured Dashboard



Adding Customization and Personalization

Now that we have secured the dashboard, we will enable authorized users to customize and personalize the portlets. Customization is used to provide the default settings for all users. Personalization allows an individual user to override the default settings, but it only affects the settings for that user. The following portlets may be customized and personalized:

- ReportPortlet
- RSSPortlet

- URLPortlet
- ORWSlideshowPortlet

Note: This example uses the ORW Slideshow Portlet to demonstrate the customization and personalization features. Examples of the other portlets are provided later on.

ORW Slideshow Portlet Example - Customization / Personalization

Prerequisites

Before starting this exercise, you will need to add the following users and roles following the same steps used in the Securing the Dashboard section.

Table 9–5 Users and Roles for the ORW Slideshow Portlet Example

Role Name	Users	Description
page-viewer	aViewUser	This user may view secured pages.
page-personalizer	aPersonalizeUser	This user may personalize portlets on a secured page.
page-customizer	aCustomizeUser	This user may customize and personalize secured pages.

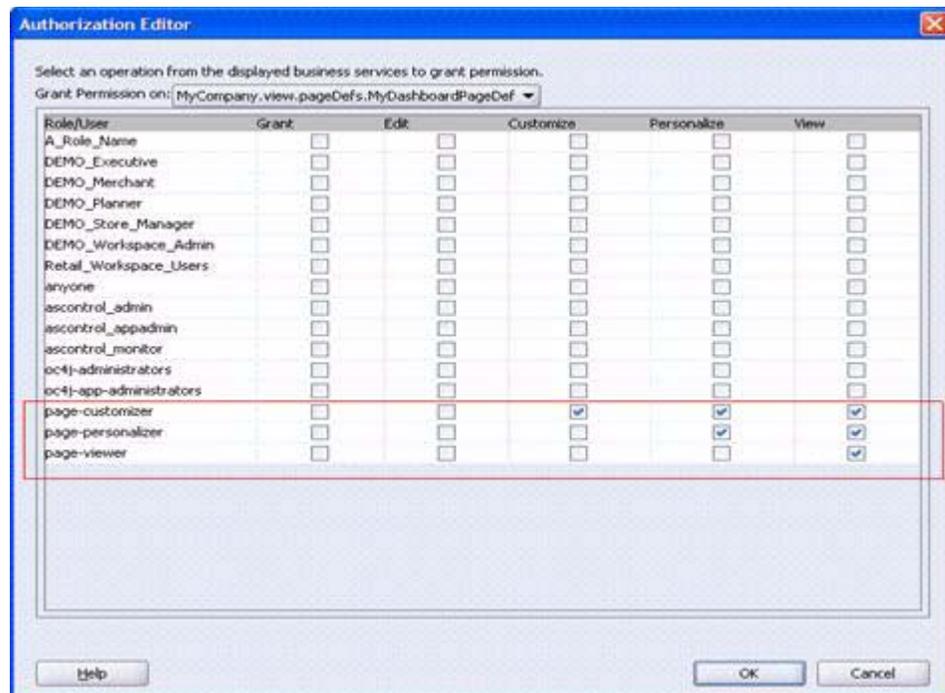
First, lets enable the portlet customize and personalize modes.

- Enable customization and personalization on the ORW Slideshow Portlet instance. In the **MyDashboard.jspx** Structure pane, select the `adfp:portlet` tag for ORW Slideshow. In the properties inspector, set the following properties:
 - `isCustomizeModeAvailable = true`
 - `isPersonalizeModeAvailable = true`

Now that the portlet allows customization and personalization, we will give the appropriate privileges to our three new roles.

1. Right click on **MyDashboard.jspx** in the Applications Navigator.
2. Choose **Go to Page Definition**.
3. In the **Structure** pane, right click on **MyDashboardPageDef**, and choose **Edit Authorization**.
4. Use the **Authorization Editor** dialog box to grant user permissions on **MyDashboard.jspx**. Select the check boxes shown below:
 - **page-customizer** – Select the check boxes under the **Customize, Personalize,** and **View** columns.
 - **page-personalizer** – Select the check boxes under the **Personalize** and **View** columns.
 - **page-viewer** – Select the check box under the **View** column.

Figure 9–39 Authorization Editor Dialog Box



The Authorization Editor enables you to choose which page actions each role/user may perform:

- **Grant** - The Grant action is not applicable for this release.
- **Edit** - The Edit action is not applicable for this release.
- **Customize** - Users with this privilege will be able to access the portlet Customization page in which the administrator can add or update various attributes. The Customization feature can be used to provide the default settings for a slideshow instance. By default, any user with Customize privileges will also have Personalize privileges.
- **Personalize** - Users with this privilege will be able to access the portlet Personalization page, in which they can add or update various attributes similar to the Customization page. The difference is that the Personalization page only affects the settings for that user, and overrides any default settings.
- **View** - Users may view the page,. If a user is not granted this permission, they will see an authorization error.

Note: When ORW is configured to use the OID LDAP security provider, an administrator grants customize and personalize privileges in Retail Workspace via the Permissions Management Administration Tool. Please refer to the section Create Permission Grants via the Permissions Management Administration Tool for further information.

5. Click OK.

Changes made through the Authorization Editor are saved in the embedded OC4J's `system-jazn-data.xml` file for immediate testing. To verify, open this file and

locate the permission for MyDashboard.jspx for the page-customizer role. You should now see the following:

```
<permission>
  <class>oracle.adf.share.security.authorization.RegionPermission</class>
  <name>MyCompany.view.pageDefs.MyDashboardPageDef</name>
  <actions>customize,personalize,view</actions>
</permission>
```

Notice that customize and personalize and view have now been added as valid actions.

Now lets take a look at ORWSlideshow portlet customization.

1. Run **MyDashboard.jspx**. Log in as aCustomizeUser. As seen below, this user now has access to customize and personalize the ORW Slideshow:

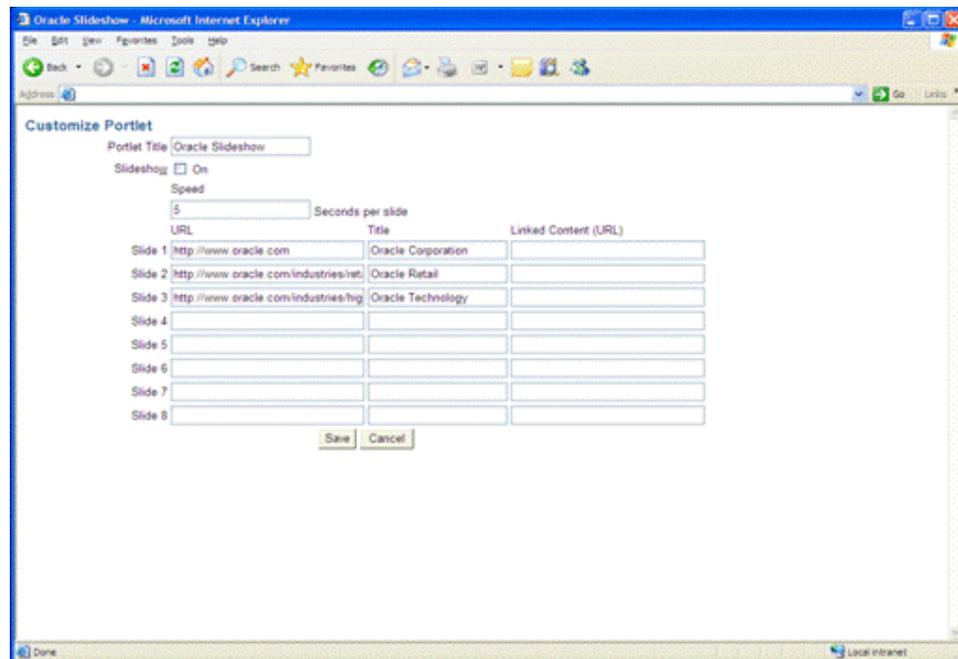
Figure 9–40 Customize and Personalize Menu Options in the Portlet



2. From the **Actions** drop-down menu, select **Customize**. You will be brought to the ORW Slideshow Portlet Customization page. From here, the user can set or change the following default values:
 - Portlet title
 - Slideshow On - yes/no
 - Slide Speed - number of seconds the slide will appear before transitioning to the next configured slide while in play mode. Valid values are positive whole numbers between 1 and 999.
 - URL - set up to eight Slide URLs
 - Title - set corresponding titles for each configured URL
 - Linked Content URL - set a related, or alternate URL for the slide URL

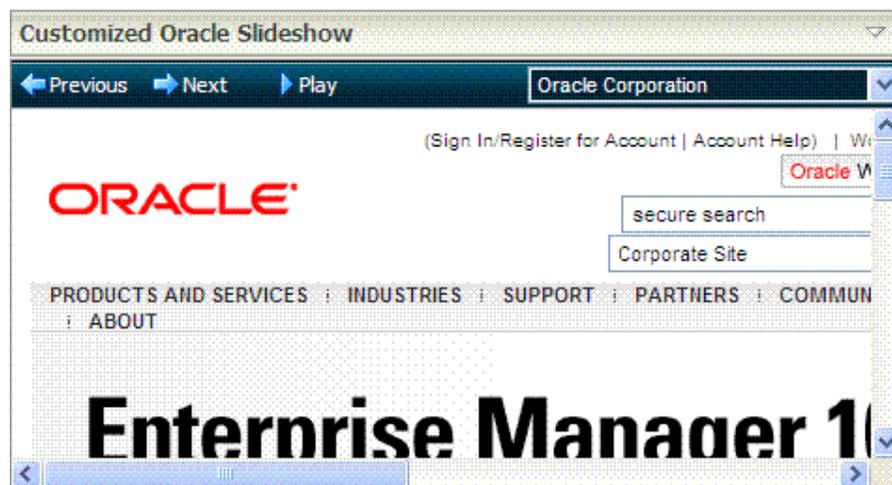
When a user accesses the page, they will see the current default ORW Slideshow configurations. It will look like the following figure:

Figure 9–41 Customize Portlet Page



3. In the **Portlet Title** field, change the portlet header to a new value, such as 'Customized Oracle Slideshow'.
4. Click **Save**.
5. The ORW Slideshow Portlet now appears with the new title. This title is now the default title for all users.

Figure 9–42 Portlet With a Customized Title



Now let's log in as a user who is allowed to personalize, but not customize, the ORW Slideshow Portlet.

1. Run **MyDashboard.jspx**. Log in as aPersonalizeUser. This user will see Customized Oracle Slideshow as the default portlet title. This user will have access to personalize the ORW Slideshow as seen below:

Figure 9–43 Personalize Menu Option in the Portlet

2. From the **Actions** drop-down menu, select **Personalize**. You will be brought to the ORW Slideshow Portlet Personalize page. From here, you can set or change the following values:
 - Portlet Title
 - Slideshow On - yes/no
 - Slide Speed - number of seconds the slide will appear before transitioning to the next configured slide while in play mode. Valid values are positive whole numbers between 1 and 999.
 - URL - set up to eight slide URLs
 - Title - set a corresponding title for each configured URL
 - Linked Content URL - set a related, or alternate URL for the slide URL

When a user accesses the page, they will see their current personalized ORW Slideshow configurations (or the default configurations if they have not personalized any of the fields). The Personalize page will look like the following figure:

Figure 9–44 Personalize Portlet Screen

Customized Oracle Slideshow - Microsoft Internet Explorer

Personalize Portlet

Portlet Title: Customized Oracle Slides

Slideshow: On

Speed: 5 Seconds per slide

Slide	URL	Title	Linked Content (URL)
Slide 1	http://www.oracle.com	Oracle Corporation	
Slide 2	http://www.oracle.com/industries/ret	Oracle Retail	
Slide 3	http://www.oracle.com/industries/hig	Oracle Technology	
Slide 4			
Slide 5			
Slide 6			
Slide 7			
Slide 8			

Save Cancel

3. In the **Portlet Title** field, change the portlet header to a new value, such as 'Personalized Oracle Slideshow'.
4. Click **Save**.
5. The ORW Slideshow Portlet now appears in the browser with the new title. This title now overrides the default title, but will only be seen by this user.

Figure 9–45 Portlet With a Personalized Title



Finally, let's log in as a user who is only allowed to view the page.

- Run **MyDashboard.aspx**. Log in as a **ViewUser**. This user will see the default title 'Customized Oracle Slideshow' as the portlet title. This user will not have access to the portlet action drop-down menu as shown below:

Figure 9–46 Portlet With No Menu Options for Users With Viewer Access



URL and Report Portlet Customization/Personalization Example

The URL Portlet and Report Portlet both allow users with sufficient privileges to modify the portlet title, the URL, and the alternate URL. The Customization and Personalization pages for both the URL and Report Portlet are shown below:

Figure 9–47 Customize Portlet Page for URL and Report Portlets

Customize Portlet

Portlet Title

URL

Alternate URL

Figure 9–48 Personalize Portlet Page for URL and Report Portlets

Personalize Portlet

Portlet Title

URL

Alternate URL

RSS Portlet Customization/Personalization Example

The RSS Portlet allows users with sufficient privileges to modify the RSS Feed URL. The Customization and Personalization pages for the RSS Portlet are shown below:

Figure 9–49 Customize Portlet Page for RSS Portlet

Customize Portlet

RSS Feed URL

Figure 9–50 Personalize Portlet Page for RSS Portlet

Personalize Portlet

RSS Feed URL

Importing/Exporting Portlet Customizations

By default, the ORW portlets have `<allow-export>` and `<allow-import>` set to true in the `oracle-portlet.xml`. It is important to note, that any customizations made during development will be exported with the application when it is deployed. For more information on Importing and Exporting Portlet Customizations, please refer to the Exporting and Importing Portlet Customizations section of the Administration Guide,

Deploying the Dashboard

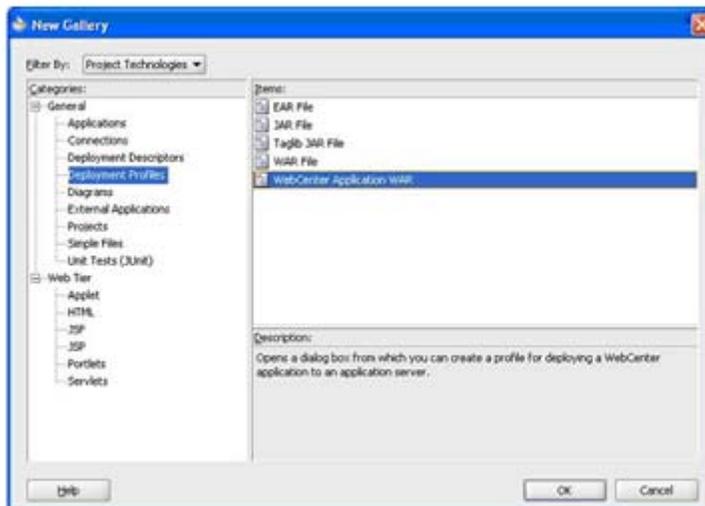
Once the dashboard has been made secure and has been tested, it can be deployed to an Oracle Application Server. The following steps to deploy the dashboard are explained in this section:

- Create a deployment profile in JDeveloper.
- Edit the Pre-configured OC4J's `system-jazn-data.xml`
- Deploy the dashboard to a generic EAR.
- Copy the EAR to the destination OAS host or shared network drive.
- Create the targeted Dashboard EAR.
- Deploy the EAR.

Creating a Deployment Profile

1. In the Applications Navigator, right-click the **ViewController** project and select **New** from the context menu.
2. In New Gallery under Categories, expand the **General** node and select **Deployment Profiles**.
3. In New Gallery under Items, select **WebCenter Application War**.
4. Click **OK**.

Figure 9–51 New Gallery Window



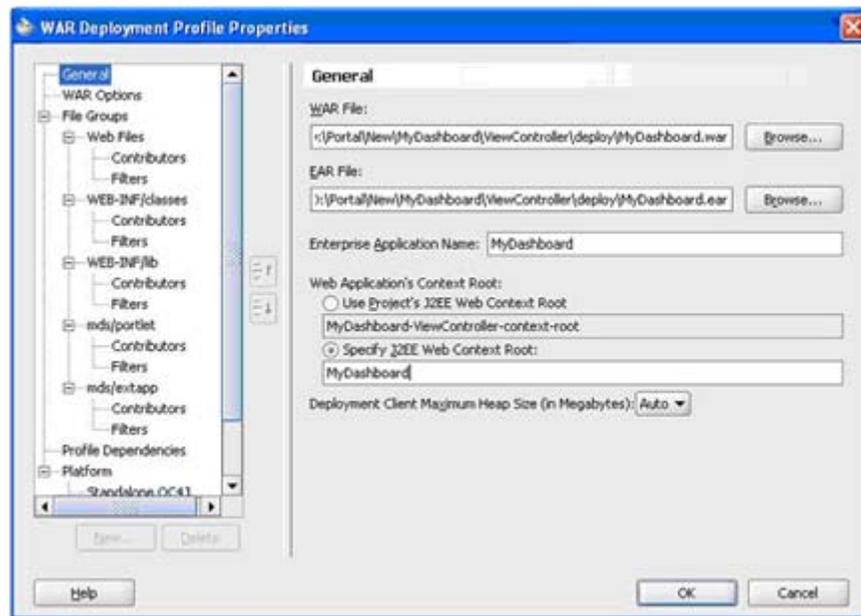
5. In the File Name field enter the name of the deployment profile (for example, MyDashboard.deploy), allow the directory name to default, and click **OK**.

Figure 9–52 Create WebCenter Application Deployment Profile Dialog Box



6. In the Applications Navigator, open the Resources folder of the ViewController project, right-click the newly created deployment profile, and select properties from the context menu.
7. Select the **Specify the web context root** option and enter a value for the field, for example, MyDashboard.

Figure 9–53 WAR Deployment Profile Properties Dialog Box



8. Click **OK**.

Define a Connection to the PreConfigured OC4J

Icons for starting and stopping the preconfigured OC4J display in the Oracle JDeveloper toolbar.

Icon	Description
	To start the pre-configured OC4J, click the Start WebCenter Pre-configured OC4J icon
	To stop the pre-configured OC4J, click the Stop WebCenter Pre-configured OC4J icon

The first time you start the pre-configured OC4J, a dialog tells you that the WebCenter pre-configured OC4J is not installed in the current user directory. This dialog offers you the option of installing it in the current user directory. Select **Yes**.

1. In the Applications Navigator, right-click the Dashboard application and select **New** from the context menu.
2. In New Gallery under Categories, expand the **General node** and select **Connections**.
3. In New Gallery under Items, select **Application Server Connection**.
4. Click **OK**.
5. On the Welcome page, click **Next**.
6. In the Connection Name field, enter a name for the connection (LocalOC4J).
7. In the Connection Type field, select **Standalone OC4J 10g 10.1.3**.

8. Click **Next**. On the Authentication screen, in the Username field, enter `oc4jadmin`. In the Password field enter `welcome`. Click the **Deploy Password** check box.
9. Click **Next**. On the Connection screen, in the Hostname field enter `localhost`. In the RMI Port field enter `22667`. Leave the URL Path field empty.
10. Click **Next**. Click **Test Connection** to test the connection to `localOC4J`.

Edit the Preconfigured OC4Js `system-jazn-data.xml`

Similar to permission grants added to the `system-jazn-data.xml` file of the embedded OC4J you need to edit the `system-jazn-data.xml` file of for the Pre-configured OC4J and add the correct security configuration to allow access to the dashboard. This configuration includes roles, users, and permission grants to these roles and users.

The location of the `system-jazn-data.xml` file for the Pre-configured OC4J is in the following folder:

```
<JDeveloper Install>/jdev/extensions/oracle.adfp.seededoc4j.10.1.3/j2ee/home/config  
<JDeveloper Install> being the directory where JDeveloper is installed.
```

Note: It is possible to copy the embedded OC4Js `system-jazn-data.xml` that you modified earlier to the Pre-configured OC4Js "config" directory. If you choose not to copy `system-jazn-data.xml`, make the same changes to the Pre-configured OC4J as you have already made to the embedded OC4J, as described in the "[Securing the Dashboard](#)" section.

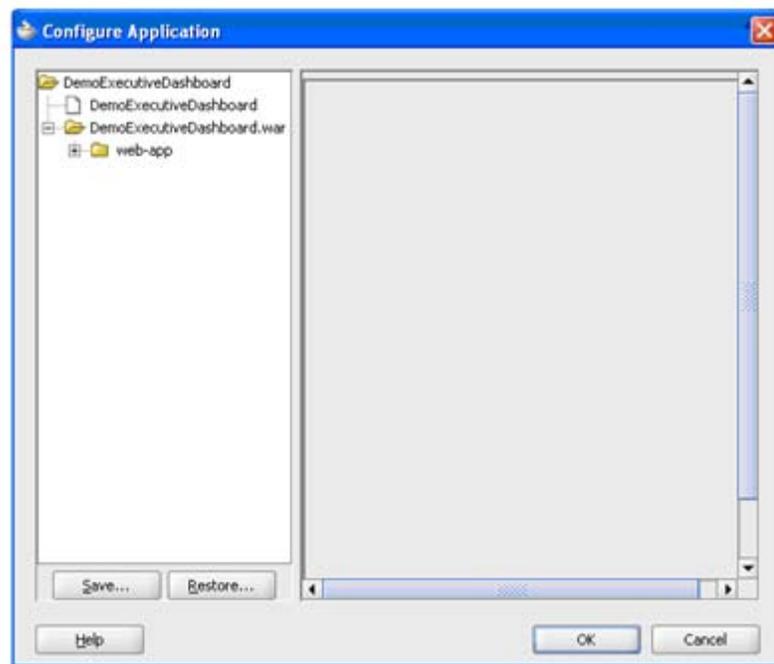
Once you have edited or copied the `system-jazn-data.xml` file of the Pre-configured OC4J you must restart the Pre-configured OC4J.

Deploy Dashboard to PreConfigured OC4J

By deploying the dashboard to the preconfigured OC4J, you are testing your deployment profile in a controlled environment.

1. In the Applications Navigator, right-click the newly created deployment profile (`MyDashboard.deploy`) and select **Deploy to** from the context menu.
2. Select the connection name for deployment that you just created. (`localOC4J`).
JDeveloper prompts you for an MDS location.
3. Specify a directory for MDS. If the directory doesn't exist, it will be created. Do not use a directory that is already in use for another dashboard. For this example, use `C:\MyDashboard-mds`

A Configure Application screen similar to the following appears.

Figure 9–54 Configure Application Screen

4. Click on **OK**.

In your deployment log tab you should see a deployment-finished message.

5. Once deployed, test your dashboard by entering the URL of your deployed dashboard in the browser window. The URL has a similar format to the following:

```
http://<your IP address>:6688/<web context root>/faces/<DashboardName.jspx>
```

For example:

```
http://myhost/MyDashboard/faces/MyDashboard.jspx
```

Note: If you are running the browser on the same machine as LocalOC4J, you may specify "localhost" instead of your IP address.

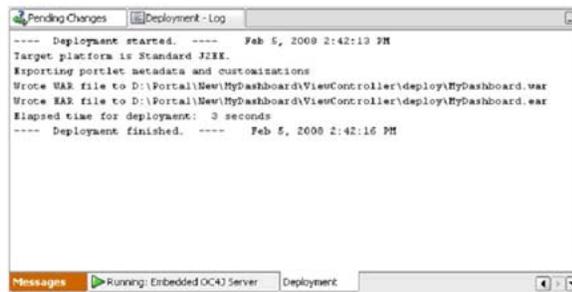
Deploy the Dashboard to Generic Ear

In the following steps you will create the generic EAR. In a later section of this chapter you will run a tool to convert this EAR to one that can be deployed on your application server.

1. In the Applications Navigator, right-click the deployment profile created above (MyDashboard.deploy) and select **Deploy to EAR file** from the context menu.
2. The deployment finished message appears in the Deployment - Log window. The log should also tell you where it placed the ear.

On Windows, the location of the EAR file is:

```
<Application root>\ViewController\deploy\MyDashboard.ear
```

Figure 9–55 Deployment - Log Window

Convert the Generic Ear to the Targeted Ear

Before deploying your EAR file, the development references contained in the file must be modified to be specific to the target instance.

Oracle Application Server with WebCenter Extensions provides a Predeployment Tool that is used to create targeted EAR files from generic EAR files. The Predeployment Tool must be run on the same host where the dashboard is to be deployed. The WebCenter Predeployment Tool is included in `portlet-client-deploy.jar`. The Predeployment Tool is executed by specifying `portlet-client-deploy.jar` in the `-jar` option of the Java command line. By default, `portlet-client-deploy.jar` and its dependencies are located in `$ORACLE_HOME/adfp/lib`, where `$ORACLE_HOME` is the Oracle WebCenter home directory.

1. FTP the file `MyDashboard.ear` to the target system on which it will be deployed.
2. Run the following command from `$ORACLE_HOME` of the OAS where the dashboard will be deployed to:

```

$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/adfp/lib/
portlet-client-deploy.jar -predeploy -source <genericEAR>
-target <targetedEAR>

```

where:

- `portlet-client-deploy.jar` is the predeployment JAR file from which the Predeployment tool is run.
- `<genericEAR>` is the generic EAR file created using Oracle JDeveloper.
- `<targetedEAR>` is the targeted EAR file, which the Predeployment tool creates during predeployment.

For example:

```

$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/adfp/lib/
portlet-client-deploy.jar -predeploy -source MyDashboard.ear
-target ./MyDashboard_Targeted.ear

```

The predeployment tool prompts for the following information:

- New MDS Repository Path on the target system. If you will be deploying multiple dashboards, you must keep MDS locations separate for each of the dashboard applications. For our example enter the following:

```

<ORACLE_
HOME>/j2ee/<oc4j-instance>/dashboards-mds-mydashboard

```

where `<ORACLE_HOME>` is the `ORACLE_HOME` environment variable set for the application server where the dashboard application is being installed, and `<oc4j-instance>` is the name of the OC4J instance you will be deploying to.

- **A New Service URL.** Since your dashboard uses portlets, the Predeployment Tool prompts for the service URL for the portlets accessible from the targeted system. The service URL has the same format as specified when creating a portlet producer.

You change the service URL in a scenario similar to the following example:

You develop the dashboard using ORW portlets that have been deployed to a development application server. Then during pre-deployment of the production version of the dashboard, you can re-target to point to an instance of the ORW portlets that has been deployed to a production application server. In that case, for **new service URL**, you specify the URL to the production deployment of the portlets.

If you do not need to specify a new service URL, that is, use the portlet producer that was used during development, click **Enter** to leave the URL unchanged.

For more information, refer to the Deploying your WebCenter Application section of the WebCenter documentation.

Deploy the Targeted EAR to OAS

The targeted EAR file is deployed to OC4J in Oracle Application Server in the production environment, using Application Server Control Console (or the Enterprise Manager).

Refer to section 12.2.3 *Deploying Your WebCenter Application Using Application Server Control Console* of the Oracle WebCenter Framework Developer's Guide included in the Oracle WebCenter documentation.

Deploying a Dashboard with Oracle Single Sign-On

Oracle Single Sign-On is a Web technology. It is implemented in the HTTP Server that typically front-ends an application server. The OC4J application servers distributed with JDeveloper do not have an associated HTTP Server. However, HTTP servers are bundled with the standard Oracle Application Server 10g.

Using the OIM Security Provider

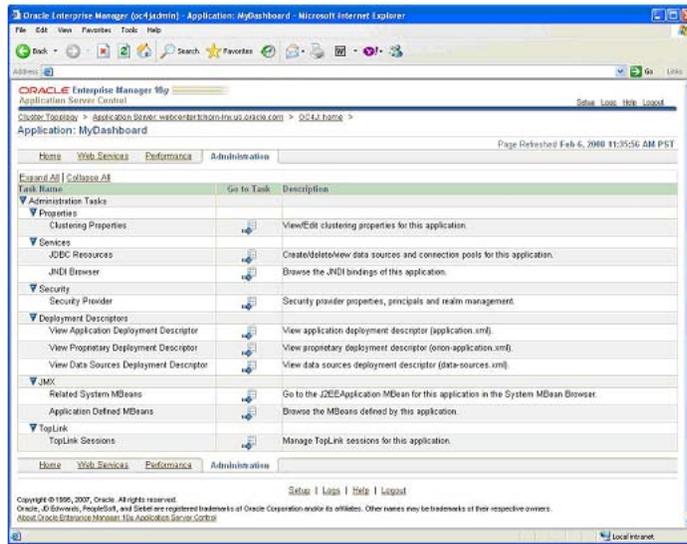
The OC4J instance where the dashboard is deployed must have an association with the OID LDAP associated with the Oracle Identity Management server. This association is documented in the *Oracle Application Server Administration Guide*.

Changing the Security Provider

After the targeted EAR has been deployed, you can switch to the Oracle Identity Management security provider.

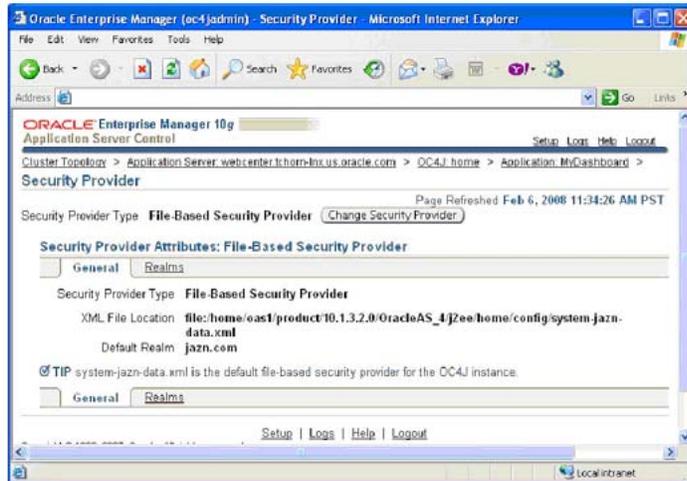
1. Using the Enterprise Manager, select the dashboard application. Then select the **Administration** tab. A window similar to the following figure should appear:

Figure 9-56 Administration Tab in the Oracle Enterprise Manager

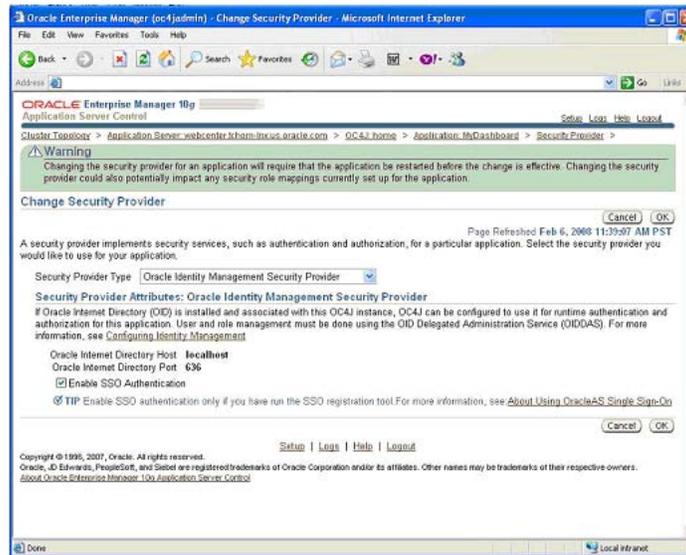


2. In the Security Provider row, select **Go to Task**. The following window appears:

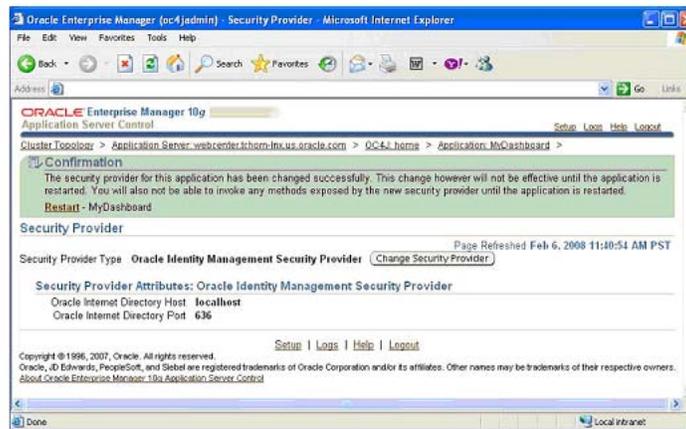
Figure 9-57 Security Provider Screen



3. Click **Change Security Provider**. In the new window, select **Oracle Identity Management Security Provider** from the Security Provider Type menu. Click **Enable SSO Authentication** to ensure the application will leverage OSSO for authentication purposes.

Figure 9–58 Change Security Provider Screen

4. Click **OK**. You are returned to the previous window, with a confirmation message:

Figure 9–59 Security Provider Screen With Confirmation of the Change

5. Click **Restart** in the confirmation message. In the next window that appears, confirm in order for the application to use OSSO.

Note: It is also possible to change to the Oracle Identity Management security provider during the deployment process.

Adding Users, Roles, and Permissions to OID LDAP

Note: While the dashboard has been associated with OID at this point, the referenced roles and permission grants to these roles must also exist in the associated LDAP. Otherwise, your dashboard will be a secure application that nobody is allowed to access. This section provides more information on creating users, roles, and permissions in the OID LDAP server.

Creating users and roles is documented in the Delegated Administrative Services (DAS) application. In DAS, a Group with public visibility is basically the same as a role. Hence, any role referenced in the deployment descriptors of the dashboard must also be created in the OID LDAP server.

Creating permission grants within the LDAP is a little more difficult. The easiest way to do this is to use the ORW Permissions Management screen available to administrators. See "[Create Permission Grants via the Permissions Management Administration Tool](#)". The user creating the permissions must have the correct permissions to do so within the LDAP security framework. Typically, this means the user ID is a member of the IAS administrator group or the JAZN administrators group.

There are other options for creating the required permission grants:

- Generate an LDIF script using the JAZN migration tool. See the *ADF Development Guide*.
- Use the OC4J jaznadmin command. See the *OAS Administrators Guide*.

Adding your Dashboard to ORW

If your deployed dashboard is not accessible to the ORW application, it must be deployed to an Oracle application server that ORW can access. Refer to "[Deploying the Dashboard](#)" for more details.

In order for the dashboard to be launchable from within ORW, it needs to be configured in the ORW `retail-workspace-page-config.xml` file. One purpose of this file is to define all of the entries in the navigation panel of the ORW application. This file can be found at

```
$ORACLE_HOME/j2ee/<oc4j-instance>/RetailWorkspace/retail-workspace-page-config.xml
```

where `<oc4j-instance>` is the name of the OC4J instance where ORW is deployed.

The `retail-workspace-page-config.xml` file is a hierarchical file containing work lists and work items. A work list contains one or more work items and each work item may contain additional work items. Work items may be defined via the `<work-item>` element, or via the `<secure-work-item>` element.

Before a user can see a `<secure-work-item>` in the navigation pane, a permission grant must be created that allows the user access. Permission grants are typically made to roles or OID Groups and not specific users. Permission grants for `<secure-work-item>` elements use the `oracle.retail.portal.security.permission.WorkElementPermission` class. However, for `<work-item>` elements, no permission grants are needed to see a link to the item in the ORW navigation pane.

Furthermore, if a dashboard has been made secure, another permission grant must be made for the user to actually view the dashboard because users may attempt to view

the dashboard outside of the ORW application. ADF uses another permission class, `oracle.adf.share.security.authorization.RegionPermission`, that must be used to grant permission to actually view a secured dashboard.

An example of an unsecured work item is shown below:

```
<work-item id="MyDashboard"
    display-string="MyDashboard"
    rendered="true"
    launchable="true"
    show-in-content-area="true"
    target-frame="_iframe" >

<url>http://myHost.myDomain.com/MyDashboard/faces/MyDashboard.jspx</url>
</work-item>
```

There are a few things to note about this entry:

- Externally deployed dashboards use the target-frame of **_iframe**. This attribute specifies that the dashboard will be rendered within an iframe.
- The show-in-content-area attribute is set to true, which means the dashboard will be rendered in the central content area of the ORW application. No more than one dashboard is displayed at a time within the content area.
- Any user will be able to access this work item, including those that have not been authenticated, as long as they have permission to view the work item's parents.

An example of a secured work item is shown below:

```
<secure-work-item id="MyDashboard"
    display-string="MyDashboard"
    rendered="true"
    launchable="true"
    show-in-content-area="true"
    target-frame="_iframe" >
    <url>http://myHost.myDomain.com/MyDashboard/faces/MyDashboard.jspx</url>
    <custom-attributes>
        <custom-attribute name="adf-permission-target">
            <value>MyCompany.view.pageDefs.MyDashboardPageDef</value>
        </custom-attribute>
    </custom-attributes>
</secure-work-item>
```

This entry is basically the same as the unsecured work item, except for the element name (`secure-work-item`) and the `<custom-attribute>` element named `adf-permission-target` which is used to secure the work item. However, when ORW processes the `retail-workspace-page-config.xml` file, the node in the navigation pane representing this secure work item will only be displayed to the user if the user, or the roles that the user is a member of, have been granted permission via a `WorkElementPermission` grant.

The ID attribute (`id="MyDashboard"`) must be configured in the `<secure-work-item>` and must be unique within the file. The Permissions Management tool uses the ID attribute to assign retail workspace permissions. The work item ID attribute is also used when assigning home pages to roles. For more information refer to ORW documentation.

The purpose of the `adf-permission-target` custom attribute element is to flag to the Permissions Management tool that this is a secured ADF page and an ADF permission grant must be specified to control access to this page. The `<value>` element of the `adf-permission-target` custom attribute specifies the permission grant target, which

must be the JSPX page's associated page definition file specification. To determine the correct value to configure for `adf-permission-target`, open the dashboard page definition file in JDeveloper. The value to use in the `<value>` element is found via a concatenation of the `Package` and `id` attributes in the root element of the page definition.

```
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
  version="10.1.3.41.57" id="myDashboardPageDef"
  Package="myproject.view.pageDefs">
```

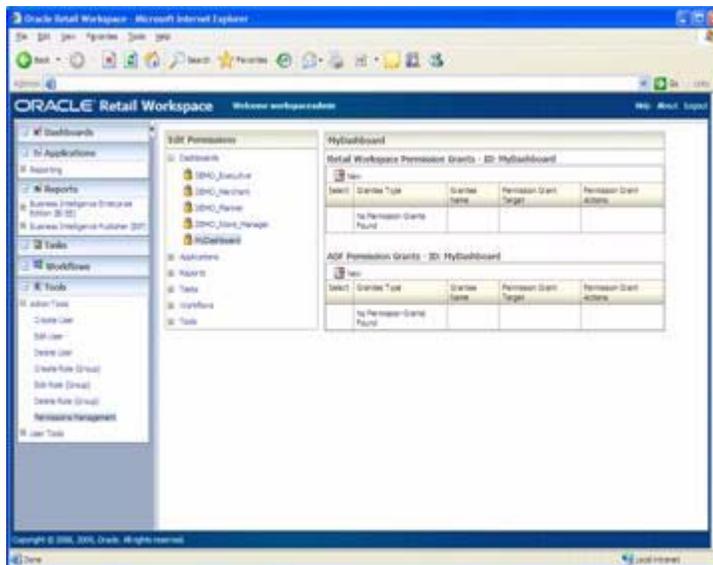
Once the secure work item has been defined in the `retail-workspace-page-config.xml` file, an administrator can then create permission grants in the OID LDAP via the Permissions Management administration tool found in the ORW application.

Create Permission Grants via the Permissions Management Administration Tool

ORW has a number of tools available to administrators. One of these tools is the Permissions Management page. When ORW is configured to use the OID LDAP security provider, the Permissions Management page is used to manage permissions for all `<secure-work-item>` elements in the `retail-workspace-page-config.xml` file. Of course, in order to access this tool one must have the correct permissions.

The following figure shows what an entry looks like for a secure dashboard defined in the `retail-workspace-page-config.xml` file with no existing permission grants. ORW Permission Grants grant access to view links to the dashboard via the left navigation pane. The ADF Permission Grants are needed because a user may attempt to access the dashboard outside of the ORW application, for example by typing in the dashboard URL directly into a browser. The ADF infrastructure checks for these permission grants on secured pages regardless of how the page was accessed. Typically, one would grant the same access to the same groups or users.

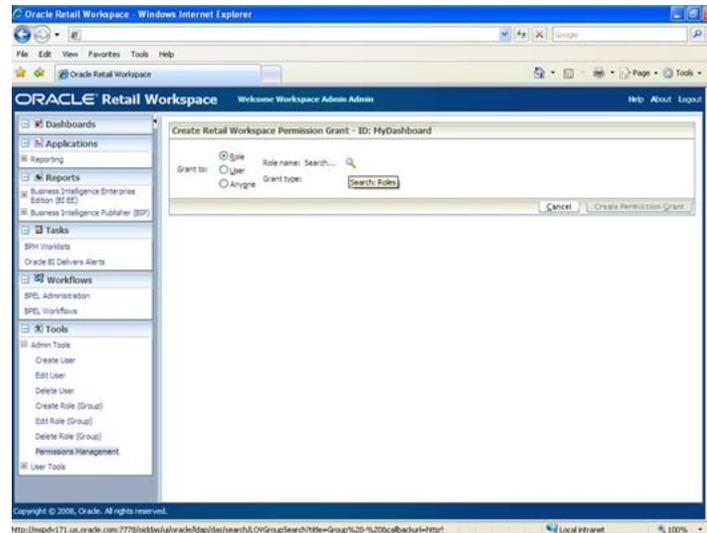
Figure 9-60 Permission Management Screen for a Secured Dashboard



1. Select the MyDashboard node in the Dashboards sub-tree of the Edit Permissions panel.

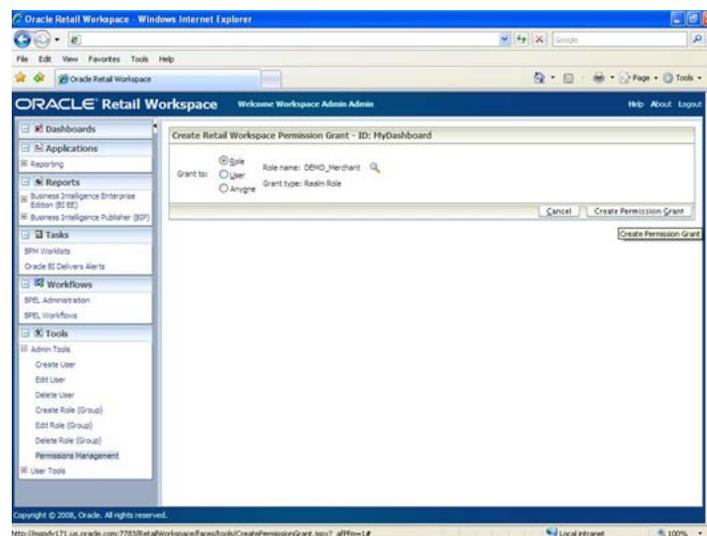
- To create a new grant, click **New Grant** above the Select column. The following page appears:

Figure 9–61 Create Retail Workspace Permission Grant Page



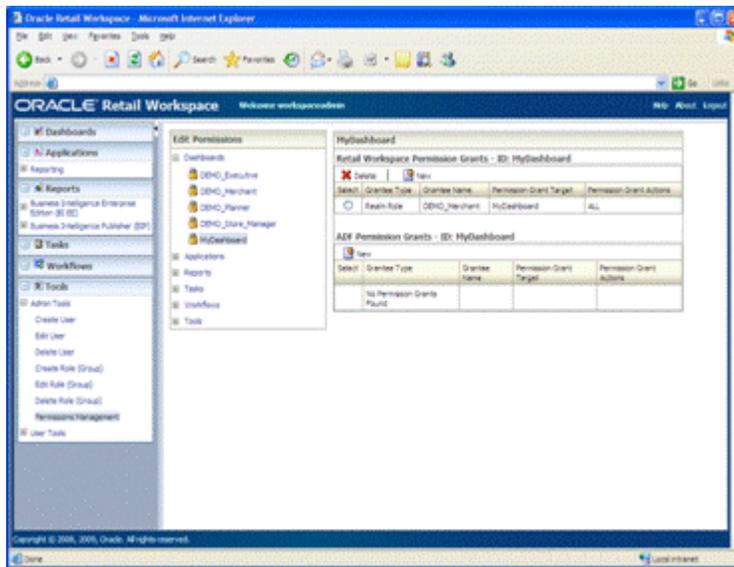
- Choose the type of Grantees. In most cases, you create a grant to a specific role.
- Click **Search** to specify the specific Realm Role. This brings up a DAS dialog.
- Select the role allowed access to MyDashboard. Once the role has been selected, control is returned to this page and the **Create Permission Grant** button is enabled.

Figure 9–62 Create Permission Grant Button Enabled



- Click **Create Permission Grant** to create the Permission Grant in the OID LDAP server and return to the main Permissions Management screen.

Figure 9–63 Permissions Management Screen

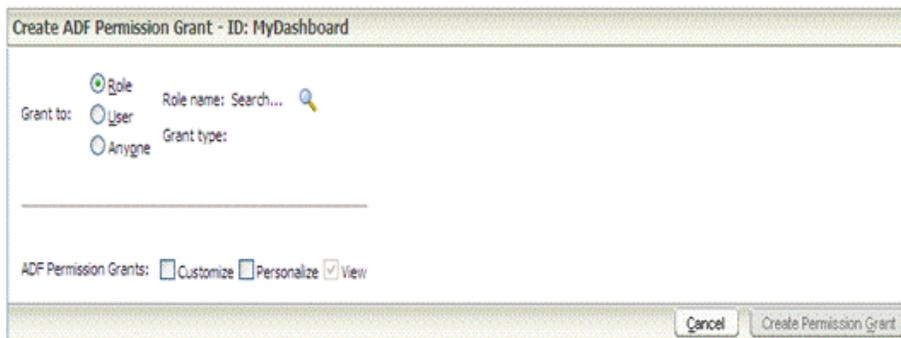


- As mentioned earlier, ADF customization and personalization privileges can be granted for secured dashboard pages via the Permission Management tool. Select the **MyDashboard** node in the Dashboards sub-tree of the Edit Permissions panel.

Note: Customization and Personalization are only available for ADF Permission Grants, and do not apply to Retail Workspace Permission Grants.

- Click **New Grant** above the **Select** column in the ADF Permission Grants table. The following window appears in the content area:

Figure 9–64 Create ADF Permission Grant Page



The administrator has the option of granting customization and/or personalization privileges to a role or user for secured dashboard pages. If a role/user is given Customize privileges, they will also be granted Personalize privileges as well. By default, the View check box is always selected and disabled. If 'Anyone' is selected, the Customize and Personalize check boxes are cleared and disabled as shown in the following figure:

Figure 9–65 Customize and Personalize Check Boxes Cleared

Create ADF Permission Grant - ID: MyDashboard

Role
Grant to: User Grant type: ADF Anyone Role
 Anyone

ADF Permission Grants: Customize Personalize View

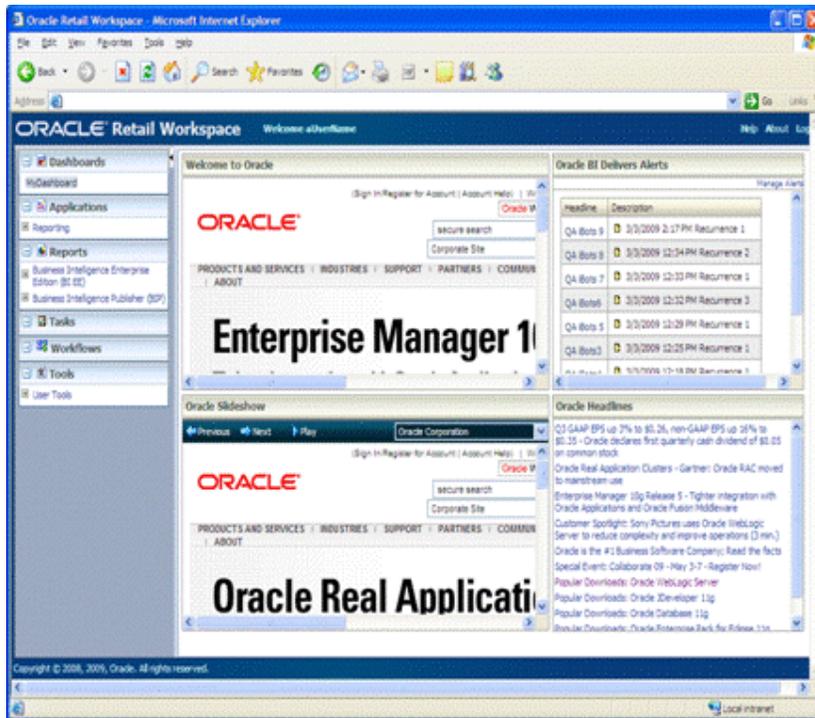
Cancel Create Permission Grant

9. Follow the same procedure as above to search, select, and create permission grants for a role or user, with the additional options of granting customize and personalize privileges.
10. Run the ORW application.
11. Log in as a user assigned to the role that has been given permission to view and access MyDashboard. Click on the **MyDashboard** link in the navigation pane on the left. If the work item and permissions have been configured correctly, MyDashboard appears in the navigation pane, and the dashboard appears in the content area when the tree node for the work item is clicked.

Note: When you use the Permissions Management tool to change permission grants, changes may not take effect immediately. This is because of the application server caching.

For the new permission grants to take effect, you may have to stop and restart the application server where the dashboard is deployed.

Figure 9–66 Dashboard Screen



Passing Parameters to a Dashboard

In previous lessons, when you created your dashboard, you hard-coded the value of several portlet parameter variables in the page definition file. You also hard-coded portlet attributes in the dashboard's JSPX file. In this lesson you will learn:

- How to change the `retail-workspace-page-config.xml` to pass parameters to the dashboard in the query string of a dashboard URL.
- How to change the dashboard page definition file to use EL expressions to retrieve query string parameters, and use those parameter values to set the `DefaultValue` of portlet parameters.
- How to change the dashboard JSPX page to use EL expressions to retrieve query string parameters and use those parameter values to set the text attribute of the `adf:portlet` tags.

Note: If you do not intend to pass URL query string parameter values to the dashboard, you may skip this lesson.

Convert Portlet Page Definition Parameter Variables

In previous lessons, the values for portlet parameter variables were hard-coded in the `pageDef.xml`. It is possible to use JSF value binding expressions to dynamically retrieve a value from a message bundle, a JSF managed bean, or an EL implicit object. One such implicit object is the `param` object, which retrieves a request parameter by name.

In the following steps, you will modify the `MyDashboard` page definition file to replace hard-coded variable values with EL expressions that retrieve the value dynamically from request parameters.

It is highly recommended you perform the parameterizing process on one portlet and one parameter at a time.

Note: The Internet Explorer browser has a roughly 2K character limit for URLs. Passing too many parameters to the dashboard will cause the combined URL (the URL to the page plus parameters) to exceed the limit. If the 2K limit is exceeded, the content area of the page will go blank and will remain blank when you click on the dashboard link in the navigation pane.

Due to this limitation, it is recommended that any page consuming the ORW Slideshow Portlet keep the parameters hard-coded in the `pageDef.xml`, rather than passing them to the dashboard page. Alternatively, a user can configure the portlet with no parameters and then use the portlet "Customize" option in development to set the default values.

1. Right-click on the `MyDashboard.jspx` page in the Application Navigation pane and click **Go to Page Definition**.
2. In the Structure pane of the page definition file, expand **executables** and then **variables**.
3. To parameterize a variable, change the `DefaultValue` in the Property Inspector to an immediate EL expression (that is, an EL expression beginning with "\$") that references a parameter by name using the EL implicit param object. Deferred EL expressions (EL expressions beginning with "#") may also be used in the page definition file.

For example, for the top report portlet `URL_TO_SHOW` parameter variable, change the `DefaultValue` from `http://www.oracle.com` to `${param.TOP_REPORT_URL_TO_SHOW}`.

Note: To locate the **variable** element associated with the `URL_TO_SHOW` parameter, first find the **parameter** element with the name equal to `URL_TO_SHOW` in the page definition file. The **pageVariable** attribute of the element specifies the **variable** element by name.

The `TOP_REPORT_URL_TO_SHOW` is a parameter that will be set in the `retail-workspace-page-config.xml` file in the steps that follow.

Note: Make sure there are no trailing spaces after the parameter.

The Top report Portlet also has an alternate URL link that should be parameterized after you parameterize the `URL_TO_SHOW` variable. Additionally, if you intend to change the default text for `ALT_URL_LABEL` you may also want to parameterize that variable.

4. Modify the ORW configuration file to add the `TOP_REPORT_URL_TO_SHOW` parameter to `MyDashboard`'s `secure-work-item`. This parameter will be passed as a query string parameter in the dashboard URL.

```
</secure-work-item>
  <secure-work-item id="MyDashboard"
```

```

        display-string="MyDashboard"
        rendered="true"
        launchable="true"
        show-in-content-area="true"
        target-frame="_iframe" >
<url>http://<yourHost>:<port>/MyDashboard/faces/MyDashboard.jspx</url>
<parameters>
  <parameter name="TOP_REPORT_URL_TO_SHOW">
    <value>http://www.oracle.com</value>
  </parameter>
  <parameter name="TOP_REPORT_ALT_URL">
    <value>http://www.oracle.com</value>
  </parameter>
</parameters>
<custom-attributes>
  <custom-attribute name="adf-permission-target">
    <value>MyCompany.view.pageDefs.MyDashboardPageDef</value>
  </custom-attribute>
</custom-attributes>
</secure-work-item>

```

Note that if your URL link contains special characters like:

http://<host>:<port>/analytics/saw.dll?Go&Path=/shared/Paint Demo/Standard Reports/Geographic Sales Summaries/Share of District Sales

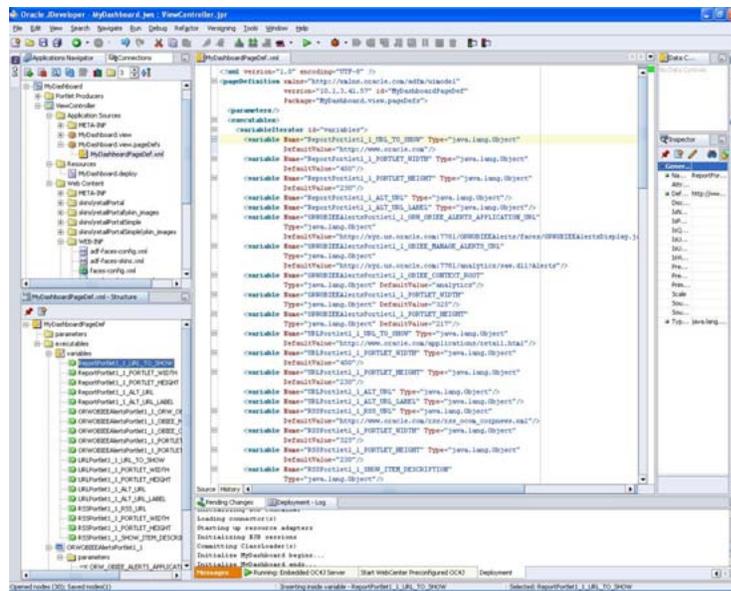
The value in the configuration file should be specified in a CDATA block, for example:

```

<value><![CDATA[http://<host>:<port>/analytics/saw.dll?Go&Path=/shared/Paint
Demo/Standard Reports/Geographic Sales Summaries/Share of District
Sales]]></value>

```

Figure 9-67 ORW Configuration File in JDeveloper




```
</managed-bean>
```

Note: The DecodeRequestParamBean works around a known issue with `HttpServletRequest.getParameters()` incorrectly decoding parameters as if they are ISO-8859-1 when in fact they are a different encoding (UTF-8 in ORW). To make use of the `DecodeRequestParamBean`, substitute EL expressions that reference the bean rather than using the JSF's implicit 'param' attribute.

This bean also works around the issues of query string parameters getting dropped from the request URLs. In the case of dashboards consuming portlets that can be customized or personalized, when the portlet mode is changed from View to Edit or Edit Defaults portlet mode and then back to the View portlet mode, the Oracle portlet container does not include the original URL's query string parameters. To work around this issue, configure the `DecodeRequestParamBean` to have session scope. The bean must be configured to have request scope for all other cases.

2. In the structure pane of the jsp page, select the `adfp:portlet` tag.
3. Right-click and select Properties from the context menu.
4. In the Portlet Properties dialog, locate the "text" property in the common properties tab and click **Bind**.
5. Enter the following in the Expression field:

```
#{decodeParam.<parameter name>}
```

For example, if the portlet tag is for the Top Report portlet, name this parameter `TOP_REPORT_TITLE` and enter `#{decodeParam.TOP_REPORT_TITLE}` in the Expression field.

6. Add this parameter to the MyDashboard secure-work-item in the ORW configuration file.

Example:

```
<parameter name="TOP_REPORT_TITLE">
<value>Welcome to Oracle</value>
</parameter>
```

7. Re-deploy the dashboard and bring it up in ORW.

Note: The value specified for the portlet title in the ORW configuration file may be an EL expression that retrieves a string from a resource bundle. For more details, see "[Internationalizing a Dashboard](#)".

References

WebCenter documentation can be accessed at:

<http://www.oracle.com/technology/products/webcenter/documentation.html>

Internationalizing a Dashboard

This section will cover the following topics:

- How to internationalize a dashboard.
 - How to internationalize strings within the dashboard application.
 - How to internationalize dashboard work item display strings and parameters in the ORW page configuration file.

In this lesson, you will perform the following steps:

- Configure the dashboard application to support locales.
- Internationalize the dashboard application. This is optional.
 - Create a resource bundle in the dashboard application. This is optional.
 - Internationalize the dashboard page. This is optional.
- Internationalize the dashboard within the ORW application.
 - Create a dashboard resource bundle for use in the ORW configuration file.
 - Modify the ORW configuration file to reference strings from the dashboard resource bundle.

Configure the Supported Locales for the Dashboard Application

ORW uses the browser locale to determine which language to use. Since the dashboard is a JSF application, you will configure the supported locales in the application `faces-config.xml` file. JSF reads the `Accept-Language` value of the HTTP header and then finds the best match from the supported locales for the application.

1. In the Applications Navigator, expand the ViewController project of the MyDashboard application. Expand the Web Content folder. Expand the WEB-INF sub-folder. Double-click `faces-config.xml` to open the file in an editor pane.
2. Register the locales you intend to support with the application. To do this, modify the `faces-config.xml` file to add `<supported-locale>` elements inside the `<locale-config>` element for each locale you wish to support. The `<locale-config>` element is a child of the `<application>` element.

The following example configures the locales that the browser application supports. This list is similar to the list of locales that ORW supports.

```
<locale-config>
  <default-locale>en</default-locale>
  <supported-locale>en_AU</supported-locale>
  <supported-locale>de</supported-locale>
  <supported-locale>fr</supported-locale>
  <supported-locale>es</supported-locale>
  <supported-locale>ja</supported-locale>
  <supported-locale>ko</supported-locale>
  <supported-locale>ru</supported-locale>
  <supported-locale>zh_TW</supported-locale>
  <supported-locale>zh_CN</supported-locale>
  <supported-locale>pt</supported-locale>
  <supported-locale>it</supported-locale>
</locale-config>
```

Internationalizing your Dashboard Application

In previous sections, you created a dashboard that consists of Reports Portlets and RSS Feed Portlets. It is also possible to add other portlets such as the URL Portlet to your dashboard.

In the case of the Report and URL Portlets, one string that you will typically internationalize is the Report/URL Portlet title. Additionally, if you configured the `ALT_URL_LABEL` parameter of the ReportPortlet or the URLPortlet, then you may also wish to internationalize the parameter value.

Note: If you have configured ORW and the dashboard to pass the portlet title and other string variables into the dashboard as URL parameters, you should skip this section and proceed to the steps for internationalizing dashboard strings within the ORW configuration file.

Create a Resource Bundle for the Dashboard Application

In the following steps, you will create a base resource bundle that contains the MyDashboard text strings.

1. In the Application Navigator, right-click on **ViewController** and click **New** to open the New Gallery.
2. In the Categories tree, select **Simple Files**, and in the Items list, select **File**.
3. In the Create File dialog, enter a name for the file, using the `.properties` extension. For this example, name it `MyDashboardMessages.properties`. We want this bundle to be in the `com.mycompany.i18n` package, and we want all bundles in a `bundles` directory. Append `\bundles\com\mycompany\i18n` to the directory name. This path is relative to your dashboard application's ViewController directory. Click **OK** to create and open the `MyDashboardMessages.properties` file.

Note: When you create a localized version of the base resource bundle, it must reside in the same Java package, that is, the same directory as the base file.

4. It is important that you add the `bundles` directory that you just created to the classpath.
 - a. Select **ViewController** in the Applications Navigator.
 - b. Right-click and select **Properties** from the context menu.
 - c. In the Java Content pane, click **Add**.
 - d. In the Project Properties dialog, select **Project Content**.
 - e. In the Choose Directory dialog, locate the `bundles` directory below your workspace ViewController directory. Double-click on the `bundles` directory to select it and close the dialog. The `bundles` directory should now be a new item in the Java Content list.
 - f. To close the Project Properties dialog, click **OK**.
5. Edit `MyDashboardMessages.properties` to include a key/value pair for each string you plan to internationalize (you may add more strings later as needed). For

this example enter the following title for the top report to
MyDashboardMessages.properties:

```
topReportTitle>Welcome to Oracle
```

Internationalize the Dashboard Page

1. Open MyDashboard.jspx.
2. Set your page encoding and response encoding to an encoding that is appropriate for all languages that the dashboard will support. In previous chapters, we recommended setting encoding and charset to "UTF-8".

If no encoding is set, the page encoding defaults to the value of the response encoding set using the contentType attribute of the page directive.

Example: Page and Response Encoding shows the encoding for the MyDashboard.jspx page:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:af="http://xmlns.oracle.com/adf/faces"
  xmlns:afh="http://xmlns.oracle.com/adf/faces/html"
  xmlns:adfp="http://xmlns.oracle.com/adf/faces/portlet"
  xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <jsp:output omit-xml-declaration="true" doctype-root-element="HTML"
    doctype-system="http://www.w3.org/TR/html4/loose.dtd"
    doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN" />
  <jsp:directive.page contentType="text/html;charset=UTF-8" />
```

3. Load the base resource bundle onto the page using the loadBundle tag, as shown below.

```
<f:loadBundle basename="com.mycompany.i18n.MyDashboardMessages"
  var="res" />
```

For this example, insert the loadBundle tag before the **head** tag.

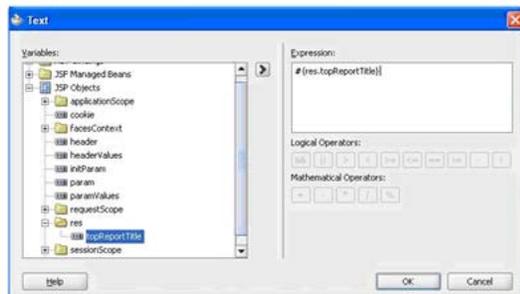
The basename attribute specifies the fully qualified name of the resource bundle to be loaded. This resource bundle should be the one created for the default language of the application. The var attribute specifies the name of a request scope attribute under which the resource bundle will be exposed as a Map. The var attribute will be used in the EL expressions that bind component attributes to a key in the resource bundle. In this example, the resource bundle will be exposed as keys under the **res** variable.

You need to load only the base resource bundle on the page. The application will automatically use the correct bundle for other locales based on the user's browser locale setting.

Note: If JDeveloper reports an error for the loadBundle tag, it indicates the specified basename is not a Resource Bundle, property file, or expression. Make sure you add the bundles directory to the project properties classpath. See ["Create a Resource Bundle for the Dashboard Application"](#).

4. With MyDashboard.jspx open and selected in the editor pane, select the adfp:portlet tag that you wish to internationalize in the MyDashboard.jspx structure pane. For this example, select the top report portlet.
5. Right click and select **Properties** from the context menu.
6. Navigate to the **Text** property and click **Bind** on the menu.
7. In the Bind to Data dialog Variables pane, expand JSP Objects. Expand the **res** variable and select **topReportTitle**. Click the right arrow to move the resulting EL expression to the Expression box. Click **OK** to confirm the binding.

Figure 9–69 Bind to Data Dialog Box - Variables Pane



This inserts the `{res.topReportTitle}` EL expression in the text property of the top report portlet's `adfp:portlet` tag.

8. Repeat steps 4 through 7 for each report or URL portlet, specifying the appropriate resource bundle key for each portlet title.

Internationalizing your Dashboard within ORW

In the following steps, you will learn how to internationalize dashboard strings configured within the ORW configuration file and how to internationalize the work item display-string attribute for your dashboard. In addition, if you have configured ORW and the dashboard to pass the portlet title and other string variables into the dashboard as URL parameters, you will learn how to internationalize those parameters.

Create a Dashboard Resource Bundle for Use in the ORW Configuration

In the following steps, you will create a base resource bundle that contains text strings for the MyDashboard application that will be used in the ORW application. Later this new resource bundle will be transferred to the application server where the ORW is deployed.

You will start by adding a new project to your application. You will use this project to keep the resource bundle distinct from resource bundles loaded by MyDashboard.jspx.

1. In the Application Navigator, right-click on the MyDashboard application and choose **New Project** to open the New Gallery. Projects should be selected under General in the Categories tree.
2. In the Items list, select **Empty Project**. Click **OK**.
3. In the Create Project dialog, enter a name for your project. Call this project `Workspace118n`. JDeveloper will provide a suitable default for the directory name. Click **OK** to create the new project.

4. In the Application Navigator, right-click on the `WorkspaceI18n` project that you just created. Select **New** in the context menu.
5. In the Categories tree, select **Simple Files**, and in the Items list, select **File**.
6. Enter a name for the file, using the `.properties` extension. For this example enter the name `MyDashboardMessages.properties`.

You want this bundle to be in the `com.mycompany.workspace.i18n` package, so append `\com\mycompany\workspace\i18n` to the directory name. This path is relative to the `WorkspaceI18n` directory of your dashboard application.

Note: When you create localized versions of the base resource bundle, these must reside in the same Java package, that is, the same directory as the base file.

Although you are giving this resource bundle the same name as the one created in the previous section, this resource bundle will be used by the ORW application.

Click **OK** to create and open the `MyDashboardMessages.properties` file.

7. Add strings to this new resource bundle. These strings will be used later to internationalize the dashboard work item display-string and optional string parameters that will be passed to the dashboard.

First, add a `myDashboard` key and its default value to `com.mycompany.workspace.i18n.MyDashboardMessages.properties`:

```
myDashboard=This is MyDashboard
```

This string will be used as the work item display string for the dashboard. This string will be displayed in the ORW navigation pane, in the dashboard work list.

If you have configured your dashboard to read the `TOP_REPORT_TITLE` query string parameter to get the value for the top report title, you must also add a `topReportTitle` key and its default value to `com.mycompany.workspace.i18n.MyDashboardMessages.properties`:

```
topReportTitle>Welcome to Oracle
```

Note that the `topReportTitle` key is not required if you did not configure your dashboard to read the `TOP_REPORT_TITLE` query string parameter.

8. At this point, you have created a resource bundle with the necessary strings to internationalize the dashboard. Now you need to copy the resource bundle to a directory that is in the classpath of the ORW application. To complete this part of the tutorial, you must also have access to the `RetailWorkspace\lang_packs` directory where the ORW application is installed:

```
$ORACLE_HOME\j2ee\instance name\RetailWorkspace\lang_packs\
```

Choose one of the following two options for placing the messages file in `RetailWorkspace\lang_packs`:

- Below the `lang_packs` directory, create a directory structure that matches the package name of the resource bundle, namely, `com\mycompany\workspace\i18n`. Copy `MyDashboardMessages.properties` to the newly created `i18n` directory.

- Package the resource bundle files in a jar file, maintaining the `com\mycompany\workspace\i18n` directory structure within the jar file. Copy this jar file to the `lang_packs` directory.

Note: To localize the dashboard, you must create versions of `MyDashboardMessages.properties` for each locale you support.

For example, create a file for each locale named `MyDashboardMessages_xx.properties`, where `xx` is the Java locale ID:

- `MyDashboardMessages_fr.properties` for French
 - `MyDashboardMessages_zh_TW.properties` for Traditional Chinese
-

After copying the resource bundles to the deployed ORW, you must stop and restart the ORW application.

Modify the ORW Configuration File to Reference Strings in the Resource Bundles

At this point, the resource bundles with the necessary strings to internationalize and localize the dashboard should have been copied to the `lang_packs` directory of the deployed ORW application. Now you must modify the ORW configuration file to make use of the resource bundles. To complete this part of the tutorial, you must have access to the deployed ORW application configuration file, `retail-workspace-page-config.xml`, which by default is located in `$ORACLE_HOME/j2ee/<instance name>/RetailWorkspace`.

1. In a text editor, open `retail-workspace-page-config.xml`.
2. Add a reference to the base resource bundle. At the top of the `retail-workspace-page-config.xml`, locate the `resource-bundles` parent element. The element will consist of one or more child `resource-bundle` elements, as follows:

```
<resource-bundles>
<!-- READ THE EXPLANATION PROVIDED IN THE FILE -->
<resource-bundle var="exampleMsgs"
resource-bundle="oracle.retail.portal.examples.i18n.Messages"/>
</resource-bundles>
```

To the `<resource-bundles>` element, add another child `resource-bundle` element that specifies the resource bundle just created:

```
<resource-bundle var="myDashboardMsgs"
resource-bundle="com.mycompany.workspace.i18n.MyDashboardMessages"/>
```

The `resource-bundle` element loads the messages in the resource bundle specified by the `resource-bundle` attribute into a Map. It then stores the Map with the name specified by the `var` attribute into JSF request scope. This allows you to use EL expressions to reference strings in the resource bundle within the configuration file.

3. Change the configuration to internationalize the dashboard work item `display-string` attribute:
 - a. Locate the `<secure-work-item>` element for the MyDashboard dashboard. You added this work item to the configuration file in a previous section.
 - b. Change the value of the `display-string` attribute from `"MyDashboard"` to `"#{myDashboardMsgs.myDashboard}"`.

`{myDashboardMsgs.myDashboard}` is an EL expression that will retrieve the value of the **myDashboard** key from the resource bundle associated with **myDashboardMsgs**, that is, `com.mycompany.workspace.i18n.MyDashboardMessages`. Evaluation of the EL expression occurs on the server during the JSF request lifecycle for the Retail Workspace JSPX page.

4. Change the configuration to internationalize any display parameters you are passing to the dashboard in the URL query string. For example, you will internationalize the top report title:

In the same `<secure-work-item>` you modified in step 3, locate the parameter named **TOP_REPORT_TITLE**. Replace the parameter value with `{myDashboardMsgs.topReportTitle}`. This is an EL expression that will retrieve the value of the **topReportTitle** key from the resource bundle associated with `myDashboardMsgs`, that is, `com.mycompany.workspace.i18n.MyDashboardMessages`.

Here is what the configuration looks like before changing the parameter value:

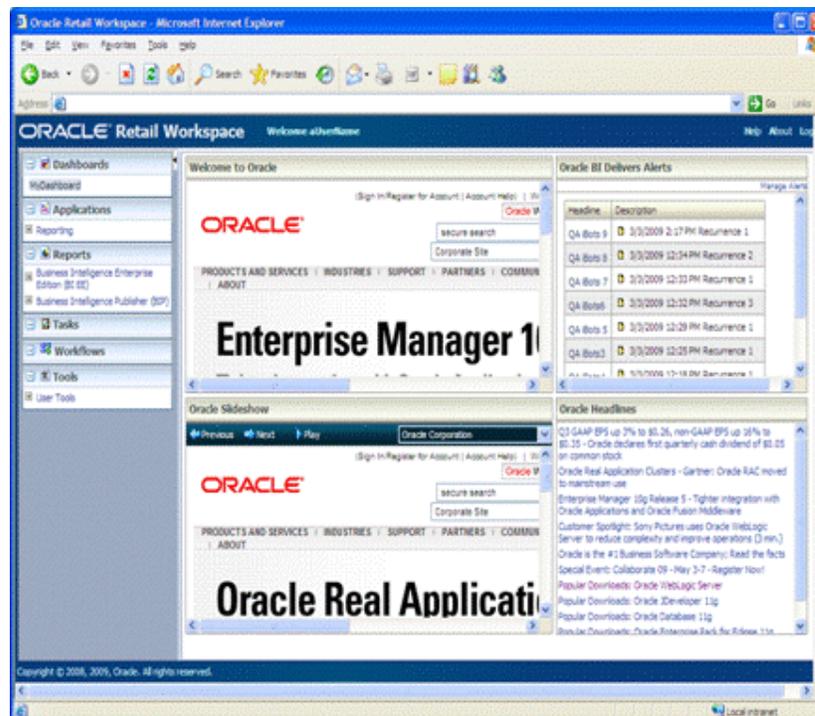
```
<parameter name="TOP_REPORT_TITLE">
<value>Welcome to Oracle</value>
</parameter>
```

After changing the parameter value:

```
<parameter name="TOP_REPORT_TITLE">
<value>#{myDashboardMsgs.topReportTitle}</value>
</parameter>
```

5. To test the changes to the ORW configuration, open up Internet Explorer and enter the URL to the deployed ORW application. Log in as a user who has access to view the dashboard.

Figure 9–70 Dashboard Screen



Summary of Configuration Changes

Below are snippets of XML showing how `retail-workspace-page-config.xml` `<resource-bundles>` and `<secure-work-item>` elements might look after making changes for internationalization:

```
<resource-bundles>
...
<resource-bundle var="myDashboardMsgs"
resource-bundle="com.mycompany.workspace.i18n.MyDashboardMessages"/>
</resource-bundles>

<secure-work-item id="MyDashboard"
  display-string="{myDashboardMsgs.myDashboard}"
  rendered="true"
  launchable="true"
  show-in-content-area="true"
  target-frame="_iframe" >
<url>http://mycompany.com:7781/MyDashboard/faces/MyDashboard.jspx</url>
<parameters>
<parameter name="TOP_REPORT_TITLE">
<value>#{myDashboardMsgs.topReportTitle}</value>
  </parameter>
<parameter name="TOP_REPORT_URL_TO_SHOW">
<value>http://www.oracle.com</value>
</parameter>
<parameter name="TOP_REPORT_ALT_URL">
<value>http://www.oracle.com</value>
</parameter>
</parameters>
<custom-attributes>
<custom-attribute name="adf-permission-target">
<value>MyCompany.view.pageDefs.MyDashboardPageDef</value>
</custom-attribute>
</custom-attributes>
</secure-work-item>
```

Team Development Considerations

Listed below are some things to consider for production quality source code supportability and team development.

In a multi-developer environment you should make sure that the portlets are deployed at a location accessible by all members of the development team.

Make sure that in a single dashboard application, a portlet producer is registered only once by one developer.

Refer to the *Oracle WebCenter Framework Developer's Guide*, chapter 11 -- Working Productively in Teams, for suggestions and best practices.

Logging

Always refer to the ORW application log whenever an error occurs on the screen. Refer to "Logging" in Chapter 2, "Backend System Administration and Configuration". In addition, make sure you are using Internet Explorer 6 and above.

Caching

Some solutions require the addition or modification of Permission Grants or a user's group membership. Permission Grants and a user's group membership may be cached by the OC4J application server instance. Additions, changes, or deletions to this information may not be used until the cached information has been refreshed. The refresh period is a configuration parameter in the `jazn.xml` configuration file. Stopping then re-starting the OC4J application server will cause the cache to be immediately refreshed.

Troubleshooting

Problem: LDAP error "User ID not found in LDAP. Please contact your System Administrator" appears after a user has successfully logged in via the SSO login page.

Possible Solutions:

- A cause could be bad entries in the `ldap-config.xml` file.
 - a. Check that the host, port, and SSL specifications are accurate.
 - b. Verify the application login distinguished name specified in the `ldap-config.xml` file exists within the ldap.
 - c. Verify the wallet specified in the password-location entry exists and the `ldap-user-pw` alias stores the password for the application login distinguished name. This can be done via commands such as `ldapsearch` or `oidadmin`.
- Make sure OID LDAP is available.

Problem: A "403 Forbidden" error appears when user accesses the main ORW page.

Possible Solutions:

If ORW uses an Oracle Internet directory as its LDAP server:

- The OC4J instance to which ORW is deployed has incorrect entries in the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml`. Verify this file specifies the security provider as "LDAP" with the correct host, port, and realm values. For more information, see the section *Associate OC4J with OID* in *Chapter 9, Workspace Application Installation* of the *Oracle Retail Workspace Installation Guide*.
- There is no permission grant made for the main ORW JSPX page. Rerun the `load_ldap_required_data.sh` script.

If ORW does not use an Oracle Internet Directory as its LDAP server:

- Verify that the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml` specifies a security provider of "XML" and there is a permission grant made for the main ORW JSPX page. Execute the `load_required_xml_jazn_data.sh` and the shell scripts it generates to create all of the required permission grants.

Problem: A "403 Forbidden" error appears in the content area when accessing a dashboard.

Possible Solutions:

If ORW uses an Oracle Internet Directory as its LDAP server:

- The OC4J instance where the dashboard is deployed has incorrect entries in the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml`. Verify this file specifies the security provider as "LDAP" with the correct host, port, and realm values. For more information, see the section *Associate OC4J with OID* in *Chapter 9, Workspace Application Installation* of the *Oracle Retail Workspace Installation Guide*.
- There is no permission grant made for the dashboard JSPX page. Use the Permissions Management administration tool to create an appropriate ADF Region permission grant.

If ORW does not use an Oracle Internet Directory as its LDAP server:

- Verify that the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml` specifies a security provider of "XML" and there is a permission grant made for the main ORW JSPX page. Execute the `load_required_xml_jazn_data.sh` and the shell scripts it generates to create all of the required permission grants.
- There is no permission grant made for the dashboard JSPX page. Create the correct permission grant using the `permgrant.sh` script. Ensure that the grant is created in the `system-jazn-data.xml` file associated with the OC4J hosting the dashboard.

Problem: "Authorization or Authentication Error" appears when user logs into ORW.

Possible Solutions:

- The user ID is not a member of the `Retail_Workspace_Users` group nor a member of a group contained in the `Retail_Workspace_Users` group.
- The user ID belongs to a group that does not have public visibility. Administrators must take care whenever modifying a group used in ORW to set the public visibility to "true". Using the DAS application, verify the groups the user belongs to have public visibility. Also check the `Retail_Workspace_Users` group for public visibility.

if ORW uses an Oracle Internet Directory as its LDAP server:

- The OC4J instance is not using the OID LDAP as its instance security provider. Verify that the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml` specifies the security provider as "LDAP" with the correct host, port, and realm specifications.
- The permission grant for the main ORW page has been deleted from OID. Look in the OID realm independent policy section (`cn=Permissions,cn=policy,cn=JAZNContext,cn=Products,cn=OracleContext`) for a permission grant with a target of `oracle.retail.portal.view.pageDefs.RetailWorkspacePageDef`.

If none exists, use the scripts found in the `ldap` sub-directory of the ORW installation home directory. In the `ldap_util.properties` file, set the `execute.ldap.scripts` property to "false" and re-run the script. Using the `ldapadd` command, re-load the permission grant made to "anyone" via the LDIF script, `create_anyone_grants.ldif`.

If ORW does not use an Oracle Internet Directory as its LDAP server:

- Verify that the file `$ORACLE_HOME/j2ee/<oc4j_instance>/config/jazn.xml` specifies a security provider of "XML" and there is a permission grant made for the main ORW JSPX page. Execute the `load_required_xml_jazn_data.sh` and the shell scripts it generates to create all of the required permission grants.
- The permission grant for the main ORW page has been deleted from the `system-jazn-data.xml` file. Use the `permlist.sh` script to see if the correct ADF RegionPermission grant exists for the page. If none exists, then create a new grant using the `permgrant.sh` script. Ensure that the grant is created in the `system-jazn-data.xml` file associated with the OC4J hosting the page.

Problem: The Permission Management link does not present buttons to create or delete permission grants.

Possible Solution:

If ORW uses an Oracle Internet Directory as its LDAP server:

- The `ldap-config.xml` file does not specify "OID" as the value of the `<identity-store>` element. Change the `ldap-config.xml` file to specify the correct value for OID. This may include adding additional entries in this file for OID.

If ORW does not use an Oracle Internet Directory as its LDAP server:

- The program is working as designed. Use the `permlist.sh` script or the JAZN admin tool to list permissions. Use the `permgrant.sh` script to create new grants and the `permdrop.sh` to delete permission grants.

Problem: Error "The LDAP server will not allow you to perform this operation. Please contact your System Administrator." appears when user clicks on Permission Management link in the Admin Tools worklist.

Possible Solution:

The current user ID does not have permission to perform the desired action. The user ID should be a member of the JAZNAdminGroup for managing permissions. This assignment may have to be performed using the `oidadmin` command found with the Oracle Identity Management Infrastructure server. The DN of the JAZNAdminGroup is `cn=JAZNAdminGroup,cn=Groups,cn=JAZNContext,cn=Products,cn=OracleContext`

Problem: Error "LDAP: error code 50 - Insufficient Access Rights" appears when an ORW administrator attempts to manage a user in DAS.

Possible Solution:

The current user ID does not have permission to perform the desired action. The user ID should be a member of the appropriate DAS administration group or have been assigned the permission directly within OID. This assignment may have to be performed using the oidadmin command found with the Oracle Identity Management Infrastructure server. The DN of the DAS administrators group is:

```
cn=OracleDASAdminGroup,cn=Groups,cn=OracleContext, [[ realm DN ]],
```

where [[realm DN]] is the realm's distinguished name.

Problem: The OSSO login screen appears when clicking on a dashboard work list item. The login screen may continue to appear even when a valid user ID and password have been entered.

Possible Solution:

A cause of the problem could be a missing ADF Region permission grant for the dashboard. Secured ADF pages, such as dashboards require two permission grants, a Work Element grant to see the link in the Navigation Panel and an ADF Region permission grant to physically access the page.

As an administrator, verify the `retail-workspace-page-config.xml` entry for the dashboard contains a custom attribute specification similar to the one below:

```
<custom-attributes>
  <custom-attribute name="adf-permission-target">
    <value>{{DASHBOARD_PAGE_DEFINITION_ID}}</value>
  </custom-attribute>
</custom-attributes>
```

where {{DASHBOARD_PAGE_DEFINITION_ID}} is the page definition file ID. Typically, this ID is the concatenation of the page definition package and the file name.

Once the custom attribute is correctly placed in the `retail-workspace-page-config.xml` file, log into ORW as an administrator and use the Permissions Management work item to create the permission grant for the dashboard.

Problem: A "500 Error" appears when accessing a dashboard.

Possible Solutions:

- The dashboard was not deployed correctly. The dashboard EAR was not targeted correctly or the MDS directory is not available on the deployed host. Part of the targeting process is to specify an MDS directory which must be available to the deployed application.
- The dashboard's URL in `retail-workspace-page-config.xml` is incorrect.
- The dashboard's OC4J may not be started yet or available to service URLs.

Problem: Clicking on any of the Example Dashboard links in RetailWorkspace home page gives 'Page Not Found Error' and the Enterprise Manager console hangs.

Possible Solution:

One cause of this problem is a lack of memory available to the OC4J instance hosting the four Example Dashboards (Executive, Merchant, Planner, Stores). Each dashboard consumes multiple portlets and the memory required by the dashboards becomes more than the default assigned to this OC4J instance. This makes dashboards very

slow to load and ends up in a 'Page Not Found' error being displayed in the content area of ORW. The enterprise manager console may also hang.

To verify this issue, view the log file `<dashboards_group>~<dashboards_oc4j>~<dashboards_group>~1` under `$ORACLE_HOME/opmn/logs` directory where `<dashboards_group>` is the group name where dashboards are deployed and `<dashboards_oc4j>` is the oc4j instance name.

Look in the log file for the error, 'java.lang.OutOfMemoryError: PermGen space', during the time you accessed the dashboard. If present, the `<dashboards_oc4j>` instance is out of memory space and needs more memory to run the dashboard applications. Follow the steps below to solve this problem:

1. Stop the `<dashboards_oc4j>` instance. This can be done by running the following command:

```
opmnctl stopproc process-type=<dashboards_oc4j>
under directory $ORACLE_HOME/oc4j/bin
```

2. Modify the `$ORACLE_HOME/opmn/conf/opmn.xml` as follows:

Find the entry `<ias-component id="<dashboards_group>" status="enabled">` in `opmn.xml`. Look for `<process-type id="<dashboards_oc4j>" module-id="OC4J" status="enabled">` under this entry. Under `<process-type id="<dashboards_oc4j>" module-id="OC4J" status="enabled"></process-type>` tag, look for `<module-data> </module-data>` tag.

The `<module-data> </module-data>` tag should have a sub tag `<category id="start-parameters"></category>`.

The `<category id="start-parameters"></category>` tag should have sub tag `<data id="java-options" value="-server -Djava.security.policy=$ORACLE_HOME/j2ee/<dashboards_oc4j>/config/java2.policy -Djava.awt.headless=true -Dhttp.webdir.enable=false"/>`.

Add the following lines to 'value' attribute of this tag.

```
-XX:MaxPermSize=512m -XX:PermSize=256m -ms512M -mx1024M
```

So, the final java-options tag entry would look like:

```
<data id="java-options" value="-server -Djava.security.policy=$ORACLE_
HOME/j2ee/<dashboards_oc4j>/config/java2.policy -Djava.awt.headless=true
-XX:MaxPermSize=512m -XX:PermSize=256m -ms512M -mx1024M
-Dhttp.webdir.enable=false"/>
```

3. Start the `<dashboards_oc4j>` instance. This can be done by running the following command:

```
opmnctl startproc process-type=<dashboards_oc4j>
under directory $ORACLE_HOME/oc4j/bin
```

4. Verify that the dashboards work by clicking any of the dashboard links in RetailWorkspace.

Problem: Dashboard does not display correctly in the content area of the ORW screen. An error, "The page could not be loaded for URL:<url>" may display, or the page may not be rendered properly.

Possible Solution:

Make sure the dashboard work item's "target-frame" attribute is set to "_iframe" in `retail-workspace-page-config.xml`.

Problem: Dashboard does not display in the content area of the ORW screen.

Possible Solutions:

- Make sure all servers defined in `retail-workspace-page-config.xml` in URLs, attributes, parameters, etc. are fully qualified (e.g. `server01.mycompany.com` rather than `server01`).
- Make sure the total length of the dashboard URL does not exceed the browser's limit. To check the URL size:
 - a. Right-click on the ORW screen somewhere other than the content area (e.g. the top of the screen where the logo is).
 - b. Select "View Source" from the context menu.
 - c. In the source window find: `src="` followed by the dashboard's URL (found in the ORW configuration file). Copy the text within the double-quotes and paste it in your favorite text editor where you can perform a character count.

If the count is greater than the Internet explorer's URL length limit, that is the problem. For IE6 the limit is roughly 2000 characters.

The URL should look something like the following:

```
http://servername.company.com:port/MerchantDashboard/faces/DemoMerchantDashboard.jspx?TOP_REPORT_TITLE=This+Week%27s+Top+Performers&TOP_REPORT_URL_TO_SHOW=http%3A%2F%2Fservername%3A7777%2Fanalytics%2Fsaw.dll%3FPortalPages%26PortalPath%3D%252Fshared%252FPortal%252Fportal%252FResizedTop%2BPerformers&TOP_REPORT_ALT_URL=http%3A%2F%2Fservername%3A7777%2Fanalytics%2Fsaw.dll%3FPortalPages%26PortalPath%3D%252Fshared%252FPortal%252Fportal%252FResizedTop%2BPerformers&MIDDLE_REPORT_TITLE=Open+Purchase+Orders&MIDDLE_REPORT_URL_TO_SHOW=http%3A%2F%2Fservername%3A7777%2FBIIPublisher%2FGuest%2FRMS%2F12.1dev%2FOrders%2Fopo_merch_dash%2Fopo_merch_dash.xdo%3F_xpf%3D%26_xpt%3D0%26_xdo%3D%252FGuest%252FRMS%252F12.1dev%252FOrders%252Fopo_merch_dash%252Fopo_merch_dash.xdo%26_xt%3DOPO%2520Merchant%2520Dashboard%2520Portal%2520Report%26_xf%3Dhtml%26_xmode%3D4&MIDDLE_REPORT_ALT_URL=http%3A%2F%2Fservername%3A7777%2FBIIPublisher%2FGuest%2FRMS%2F12.1dev%2FOrders%2Fopo_merch_dash%2Fopo_merch_dash.xdo%3F_xpf%3D%26_xpt%3D0%26_xdo%3D%252FGuest%252FRMS%252F12.1dev%252FOrders%252Fopo_merch_dash%252Fopo_merch_dash.xdo%26_xt%3DOPO%2520Merchant%2520Dashboard%2520Portal%2520Report%26_xf%3Dhtml%26_xmode%3D4&BOTTOM_REPORT_TITLE=Yesterday%27s+Flash+Sales&BOTTOM_REPORT_URL_TO_SHOW=http%3A%2F%2Fservername%2Fanalytics%2Fsaw.dll%3FGo%26Path%3D%252Fshared%252FPortal%252FYesterday%2527s%2520Flash%2520Sales&BOTTOM_REPORT_ALT_URL=http%3A%2F%2Fservername%2Fanalytics%2Fsaw.dll%3FGo%26Path%3D%252Fshared%252FPortal%252FYesterday%2527s%2520Flash%2520Sales&TOP_RSS_URL=http%3A%2F%2Fwww.oracle.com%2Frss%2Frss_ocom_pr.xml&MIDDLE_RSS_
```

URL=http%3A%2F%2Fwww.oracle.com%2Frss%2Frss_ocom_corpnews.xml&BOTTOM_RSS_
 URL=http%3A%2F%2Fwww.oracle.com%2Ftechnology%2Fsyndication%2Frss_otn_soft.xml

Problem: Portlet on a dashboard showing "Portlet Not Available" message.

Possible Solutions:

- Make sure all servers defined in the `retail-workspace-page-config.xml` in URLs, attributes, parameters, etc. are fully qualified (e.g. `server01.mycompany.com` rather than `server01`).
- Find out if the portlets are deployed in the same OC4J instance as ORW or the dashboards. It is strongly recommended that ORW, the portlets and the dashboards are deployed in their own OC4J instances.
- Portlets are deployed on a server with performance issues. Redeploy on a better server.
- For additional information, refer to chapter 13, "Monitoring your WebCenter Application" of the *Oracle WebCenter Framework Developers Guide*. This chapter presents diagnostic information related to failed portlet producer requests that may help in troubleshooting portlet errors.

Problem: Error is displayed within a portlet on a dashboard.

Possible Solutions:

- The URL parameter in `retail-workspace-page-config.xml` corresponding to that portlet contains special characters that are not escaped. These should be escaped properly. It may also be necessary that the URL be placed in a CDATA section. See "Showing Reports in Dashboards" in the Implementation Guide.
- The URL is invalid. To check the URL, copy and paste it into a browser window. The content should display in the browser.

Problem: Login page is displayed within a portlet on a dashboard.

Possible Solution:

The content to be displayed in the portlet is not OSSO enabled. Redeploy or reconfigure the content to be OSSO enabled.

Problem: Login page is displayed, but entering a valid user-ID and password results in an error.

Possible Solutions:

If ORW uses an Oracle Internet Directory as its LDAP server and OSSO:

- The `Retail_Workspace_Users` group or the group containing the login may have its "public visibility" set to false. Edit the group using the DAS or the `oidadmin` command to set the public visibility of this group to "true".

If ORW does not use an Oracle Internet Directory as its LDAP server:

- Review the `system-jazn-data.xml` file and verify that a `<login-module>` element exists for the Retail Workspace application or dashboard. A `<login-module>` element must exist for every application using an alternative LDAP for authentication.
- Review the application's `orion-application.xml` file found in the directory, `$ORACLE_HOME/j2ee/[oc4j instance]/application-deployments/[app-name]`. Verify that the `<jazn>` element contains the two properties listed below:

```
<property name="role.mapping.dynamic" value="true" />
<property name="custom.loginmodule.provider" value="true" />
```

Problem: Portlet not showing any content.

Possible Solutions:

- Right-click inside the portlet and then select Refresh within the context menu.
- Make sure all servers specified in the URL are fully qualified (e.g. server01.mycompany.com rather than server01).

Problem: Error is displayed in the launched browser window when clicking on an application's link in the Applications worklist.

Possible Solution:

The application's URL in `retail-workspace-page-config.xml` is invalid. To check the URL, copy and paste it into a browser window. Be sure to append to the URL any parameters specified in configuration file following the URL. For example if the URL is `http://myserver/mycontext/mypage` and a parameter follows with a name of "param" and a value of "prod", the combined URL is `http://myserver/mycontext/mypage?param=prod`.

Problem: Login page is displayed in the launched browser window when clicking on an application's link in the Applications worklist.

Possible Solutions:

- The application is capable of participating in the OSSO authentication process but is not OSSO enabled. See the application's documentation for instructions on how to OSSO enable the application.
- The application does not participate in the OSSO authentication process and has not been configured as an external application in ORW and OSSO. See [Chapter 4, "External Applications"](#) for instructions on configuring external applications in ORW and OSSO.

Problem: Right-clicking on a link in the ORW navigation pane and selecting Open in New Window from the browser context menu does not change the selection in the pane, and it does not launch that link in a new browser window. Instead, a new copy of ORW is launched.

Possible Solution:

This behavior applies to any link in the navigation panel that is configured as `show-in-content-area = "true"`, such as links in the Dashboard work list.

This also applies to the About link. Selecting Open in New Window launches a new copy of ORW, with the currently selected work item displayed in the content area.

This is the expected behavior of the ADF "command link" component used to render the link.

Problem: When multiple browser windows are open and associated with a single browser session, selecting an item in the navigation pane in one window may affect what is displayed in the other browser windows when they are refreshed.

Possible Solution:

This behavior occurs when the user opens the ORW application in Internet Explorer, logs in, right-clicks on one of the application's hyperlinks and selects Open in New Window from the browser context menu.

The ORW application opens in a new browser window. The user is logged into the application, and the dashboard that was displayed in the original window is displayed in the new browser window. If the user selects a different dashboard in either of the browser windows, and then refreshes the other window, both windows show the same dashboard.

The reason for this behavior is that the ORW application uses JSF session-scoped managed beans to maintain user interface state information in the browser session. This affects the application behavior if multiple browser windows share the same session.

User interface state information that may be affected includes:

- The role-based lists of work items to display in the navigation panel
- The currently selected work item

This is the expected behavior when multiple ORW windows share the same browser session.

To avoid this behavior, do not open multiple browser windows that share the same session.

Problem: When interacting with a tree or with buttons or links on the navigation pane of the ORW page, nodes in the trees collapse unexpectedly or disappear.

Possible Solution:

This occurs when the browser session times out because of a prolonged period of inactivity. If the Single Sign-On session also times out concurrently, tree nodes may also disappear if unauthenticated users have not been given permission to view the work items.

This is the expected behavior when the session times out. The ORW application uses JSF session-scoped managed beans to maintain user interface state information in the browser session. The application also uses the ADF Session Change Manager to track the expansion/selection state of interface components. This affects the application behavior if the browser session times out. The default ORW session timeout period is 35 minutes.

User interface state information that may be affected includes:

- The expansion state of tree nodes and containers in the navigation panel
- The role-based lists of work items to display in the navigation panel
- The currently selected work item

To prevent session timeout, users should not leave the application open for prolonged periods of inactivity. They should log out of ORW and close the browser window if they plan to be away for a prolonged period.

If it appears the session has timed out:

- The user should refresh the ORW application using the browser Refresh button.
- If at this point the user realizes he/she is no longer logged in to the application, he/she should click the Login link and log in again.

It is possible for customers to change the ORW `web.xml` deployment descriptor to increase the value of `<session-timeout>`, but this is not recommended.

Problem: When interacting with a tree, worklist, buttons or links on the navigation pane of the ORW page, the content displayed in the page content may revert to the

default home page associated with the roles to which the logged-in user has been assigned.

Possible Solution:

This occurs when the browser session times out because of a prolonged period of inactivity.

This is the expected behavior when the session times out. The ORW application uses JSF session-scoped managed beans to maintain user interface state information in the browser session. The application also uses the ADF Session Change Manager to track the expansion/selection state of interface components. This affects the application behavior if the browser session times out. The default ORW session timeout period is 35 minutes.

User interface state information that may be affected includes:

- The expansion state of tree nodes and containers in the navigation panel
- The role-based lists of work items to display in the navigation panel
- The currently selected work item

To prevent session timeout, users should not leave the application open for prolonged periods of inactivity. They should log out of ORW and close the browser window if they plan to be away for a prolonged period.

If it appears the session has timed out:

- The user should refresh the ORW application using the browser Refresh button.
- If at this point the user realizes he/she is no longer logged in to the application, he/she should click the Login link and log in again.

It is possible for customers to change the ORW `web.xml` deployment descriptor to increase the value of `<session-timeout>`, but this is not recommended.

Problem: "Error Accessing BIP Server" is shown when expanding the BIP work item.

Possible Solutions:

Make sure you are familiar with the ["Exposing Reports Links in the Navigation Panel"](#) section of [Chapter 6, "Integration Methods and Communication Flow"](#).

Analyze the error message in the application's log file.

1. If the error is related to an invalid user ID or password, verify that:
 - a. The `bipublisher-login-id` specified in `retail-workspace-page-config.xml` is declared in OID and belongs to the BIP (or XMLP) Administrator role.
 - b. The password stored in the wallet at the location specified in `bipublisher-password-wallet-location` and corresponding to the alias value in `bipublisher-password-alias` is the correct password for the ID.
 - c. The logged-in user is authorized to access the BIP server specified in the configuration file.
2. If the error is related to a permission problem, make sure 1.a above applies.
3. If the error is related to webservices, make sure the URL corresponding to `bipublisher-webservices-url` in the ORW configuration is valid. Copy and paste it into a browser window. You should get a Services screen.

4. If the error is related to a server error, make sure the URL corresponding to `bipublisher-webservices-url` in the ORW configuration is valid. Copy and paste it into a browser window. You should get a Services screen.

Problem: Error when expanding the Oracle BI EE work item.

Possible Solutions:

Make sure you are familiar with section "Exposing Reports Links in the Navigation Panel" of the Implementation Guide.

Analyze the error message in the application's log file.

1. If the error is related to an invalid user ID or password, verify that:
 - a. The `biee-login-id` specified in `retail-workspace-page-config.xml` is declared in Oracle BI EE as an Administrator or an Impersonator.
 - b. The password stored in the wallet at the location specified in `biee-password-wallet-location` and corresponding to the alias value in `biee-password-alias` is the correct password for the ID.
 - c. The logged-in user is authorized to access the Oracle BI EE server specified in the configuration file.
2. If the error is related to a permissions problem, make sure 1.a above applies.
3. If the error is related to webservices, make sure the URL corresponding to `biee-webservices-url` in the ORW configuration is valid. Copy and paste it into a browser window and append `?wsdl` to the end of it. You should get a WSDL definition screen.
4. If the error is related to a server error, make sure the URL corresponding to `biee-webservices-url` in the ORW configuration is valid. Copy and paste it into a browser window and append `?wsdl` to the end of it. You should get a WSDL definition screen.

Problem: Error when clicking on a report's hyperlink in the BIP reports section of the navigation panel.

Possible Solution:

Make sure the URL corresponding to `bipublisher-reports-url-prefix` in the ORW configuration is valid. Copy and paste it into a browser window. You should get an OSSO login screen or the BIP login screen depending on whether BIP is OSSO enabled or not.

Problem: Error when clicking on a report's hyperlink in the Oracle BI EE reports section of the navigation panel.

Possible Solution:

Make sure the URL corresponding to `biee-reports-url-prefix` in the ORW configuration is valid. Copy and paste it into a browser window and append `?Dashboard` to the end of it. You should get an OSSO login screen or the Oracle BI EE login screen depending on whether Oracle BI EE is OSSO enabled or not.

Problem: Portlet on a dashboard for an Oracle BI EE report shows "Bookmark Link Error On Windows With FAT32".

Possible Solution:

This issue occurs when Oracle BI Presentation Services is running on a Microsoft Windows machine with a FAT32 file system.

Convert the FAT volumes to NTFS. Refer to your Windows operating system documentation for instructions on file system conversion.

See Section 2.2.7 at the following URL:

http://download.oracle.com/docs/cd/E10415_01/doc/bi.1013/e10404/bi_ee_rns.htm

Problem: A new secure work item, such as a dashboard, is not visible in the Dashboard work list, even after configuring permission grants for the work item.

Possible Solutions:

- Make sure you have logged in as a user who belongs to one of the roles that has been granted permission to view and access the dashboard.
- OC4J may be using stale cached permission grants. Either wait for OC4J to clear its caches, or stop and re-start the OC4J instance where the dashboard is installed.

Problem: Permission grants have been modified (added or deleted) for a secure work item or secure work list. The logged in user does not see the expected result after changing the grants.

Possible Solution:

If ORW uses an Oracle Internet Directory as its LDAP server:

- OC4J may be using stale cached permission grants. Either wait for OC4J to clear its caches, or stop and re-start the OC4J instance where Retail Workspace is installed.

If ORW does not use an Oracle Internet Directory as its LDAP server:

- When OID is not used, permission grants are stored in the `system-jazn-data.xml` file. These grants are only read once when the OC4J is booted. Restart the OC4J instance hosting Retail Workspace.

Problem: A user who does not have permissions to personalize or customize a portlet is able to view a portlet's customization or personalization screen if using the same browser window as a previous user.

Possible Solutions:

This will occur when a user with sufficient permissions to customize or personalize a portlet accesses the portlet's personalization or customization screen and fails to exit from the screens before logging out of RetailWorkspace. Any subsequent users using the same browser window will be presented with the customization or personalization screen when they try to display the same portlet. This is a known Oracle WebCenter issue and a defect has been logged.

To avoid running into this issue, it is recommended that the users always exit the portlet personalization and customization screens before logging out of the Retail Workspace application, and close the browser after logging out.

Problem: Alerts Portlet displays the error message "The application encountered an error. Please contact your system administrator."

- Verify that the Oracle BI EE RSS Alerts URL is available. Try the following link in a browser:

`http://<host>:<port>/<analytics-context-root>/saw.dll?RssAlerts`

where `<host>` and `<port>` are the host and port numbers of the Oracle Application Server where Oracle BI EE is deployed and `<analytics-context-root>` is the context-root of Oracle BI EE Presentation Services. If Oracle BI EE is single sign-on enabled this URL should redirect to an SSO login page. Try logging in, you should

see either the RssAlerts feed if the browser has a built-in RSS feed reader or the feed XML.

- Verify that the ORWOBIEEAlerts application URL is available. Try the ORWOBIEEAlerts application URL in a browser. This application should be deployed on the same Oracle Application Server where Oracle BI EE presentation services are deployed.

```
http://<host>:<port>/<ORWOBIEEAlerts-context-root>/faces/ORWOBIEEAlertsDisplay.jspx?OBIEE_CONTEXT_ROOT=<analytics-context-root>
```

where *<host>* and *<port>* are the host and port numbers of the Oracle Application Server where the ORWOBIEEAlerts application is deployed, *<ORWOBIEEAlerts-context-root>* is the context-root of ORWOBIEEAlerts application and *<analytics-context-root>* is the context-root of Oracle BI EE presentation services. This URL should be statically protected. If the URL is protected correctly, it should show an SSO login page. Try logging in, you should be able to see your iBot alerts.

- Verify that you are setting correct values for the ORWOBIEEAlertsURL and OBIEE_CONTEXT_ROOT parameters in your Alerts Portlet.

Problem: Manage Alerts link in the Alerts Portlet displays a "page not found" error.

Possible Solution:

Verify that you have set correct value for Manage Alerts URL in the Alerts Portlet. The URL should be in the following form:

```
http://<host>:<port>/<analytics-context-root>/saw.dll?Alerts
```

where *<host>* and *<port>* are the host and port numbers of the Oracle Application Server where Oracle BI EE presentation services are deployed and *<analytics-context-root>* is the context-root of Oracle BI EE presentation services.

Problem: Oracle BI Delivers Alerts link in the ORW Navigation pane displays a "page not found" error.

Possible Solution:

Verify that you have set the correct value for the Oracle BI Delivers Alerts link in the ORW page configuration file. The URL should be in the following form:

```
http://<host>:<port>/<analytics-context-root>/saw.dll?Alerts
```

where *<host>* and *<port>* are host and port numbers of the Oracle Application Server where Oracle BI EE presentation services are deployed and *<analytics-context-root>* is the context-root of Oracle BI EE presentation services.

Problem: A user is able to view a previously logged-in user's alerts if the same browser window is used.

Possible Solution:

ORW uses Alerts published by Oracle BI EE as an RSS Feed to display a list of Alerts in the ORW Alerts Portlet. When Oracle BI EE is OSSO enabled, there is a seamless integration with Alerts. Once logged into the ORW application a user is able to view his/her own Alerts. Upon logout from ORW (and therefore out of OSSO), Oracle BI EE does not log the user out. Hence if another user logs into ORW using the same browser window he or she will view the previously logged-in user's Alerts.

A defect has been logged against the Oracle Business Intelligence Enterprise Edition 10.1.3.3.0.

The work around is to close the browser session upon logging out of the ORW application

Problem: BPMWorklists/BPELWorkflows links display the error message "Portlet unavailable" on the page in the content area of ORW.

Possible Solution:

Verify the ORWBPELPortlets URL is available. Try the URL in a new browser. This URL should be in the following form:

`http://<hostname>:<port>/<contextroot>/portlets/wsrp2?WSDL`

where *<host>* and *<port>* are host and port numbers of the Oracle Application server where the ORWBPELPortlets application is installed and *<context-root>* is the context-root of the ORWBPELPortlets application.

Problem: BPMWorklists/BPELWorkflows links display the error message "The portlet was unable to locate the workflow context for the user. Please contact your system administrator".

Possible Solutions:

- If the BPMWorklist link displays this error message then verify that the files `wf_client_config.xml` and `wf_config.xml` are copied from the `$ORACLE_HOME/bpel/system/services/config` directory of the BPEL Server installation to the `$ORACLE_HOME/j2ee/<OC4J_INSTANCE>/ORWBPELPortlets/config` directory of the OC4J instance where ORWBPELPortlets are installed.
- If the BPELWorkflow link displays this error message then verify that the file `context.properties` in the `$ORACLE_HOME/j2ee/<OC4J_INSTANCE>/ORWBPELPortlets/config` directory contains valid URLs for `bpel.server.url` and `java.naming.provider.url`.
- Ensure that the username and password for the OC4J administrator in the BPEL Server matches the username and password entered in the wallet located at `$ORACLE_HOME/j2ee/<OC4J_INSTANCE>/ORWBPELPortlets/security/bpelwallet`.
- Ensure that the OC4J administrator name stored for the BPELServer in the wallet located at `$ORACLE_HOME/j2ee/<OC4J_INSTANCE>/ORWBPELPortlets/security/bpelwallet` has administrative permissions. The administrator must be a member of the `oc4j-administrators` role and have RMI login permission.
- Ensure that the realm name stored in the wallet located at `$ORACLE_HOME/j2ee/<OC4J_INSTANCE>/ORWBPELPortlets/security/bpelwallet` is a valid realm in the BPEL Server. Check the file `is_config.xml` located in the `$ORACLE_HOME/bpel/system/services/config` directory of the BPEL Server for valid BPEL realms.
- Ensure that the realms information in the `wf_config.xml` and `is_config.xml` files located in the `$ORACLE_HOME/bpel/system/services/config` directory of the BPELServer are in sync.
- Ensure that the `BPELPM_CONNECTION_POOL` under the SOA suite OC4J instance is connecting successfully to the dehydration database. Verify this connection by using the test connection link in the JDBC Resource page of the SOA suite OC4J instance through Enterprise Manager Console.

Problem: BPMWorklists/BPELWorkflows links in the ORW Navigation pane display a "page not found" error.

Possible Solution:

Verify that you have set the correct values for the BPMWorklists/BPELWorkflows links in the ORW page configuration file. The URLs should be in the following form:

```
http://<host>:<port>/<context-root>/faces//ORWBPMWorklist.jspx
```

and

```
http://<host>:<port>/<context-root>/faces/ORWBPELWorkflow.jspx
```

where <host> and <port> are host and port numbers of the Oracle Application Server where ORWBPELPages application is deployed and < context-root> is the context-root of ORWBPELPages application.

Problem: A URL that is configured in a Report, URL, or ORW Slideshow portlets takes over the entire browser window preventing the user from accessing the portlet actions to customize and personalize.

Possible Solutions:

A Web site that sets the window.top location to window.self.location, or sets its target frame to _top or _parent, will result in that site taking over the browser window. To prevent this from happening, all URLs entered into these portlets need to be able to display correctly within an iframe. To prevent users from customizing or personalizing these portlets, set the 'isCustomizeModeAvailable' and 'isPersonalizaModeAvailable' properties for the <adfp:portlet> tag to false. If Retail Workspace is configured to use Oracle Internet Directory as the LDAP server, the customization and personalization privileges can be removed using the Retail Workspace Permission Management tool.

