

Oracle® Retail Accelerators Guide for WebCenter 11g

September 2010

ORACLE CONFIDENTIAL.

For authorized use only.

Do not distribute to third parties.

Note: The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Contents

1 Introduction

About Oracle Retail Accelerators	1-2
Accessing the Extending WebCenter Spaces White Paper	1-2

2 Customizing WebCenter Spaces

Setting Up a Development Environment	2-1
Setting Up WebCenter Resource Catalogs	2-2
Resource Catalog Overview	2-2
Setting Up a Development Environment	2-3
Customizing the WebCenter Resource Catalogs	2-3
Packaging and Deploying Customized Catalogs	2-4
Setting Up WebCenter Page Styles	2-4
Setting Up a Development Environment	2-5
Out of the Box Page Styles and Style List	2-5
Custom Page Styles	2-5
Creating a New Page Style	2-5
Setting Up the Page Style List	2-8
Packaging and Deploying the Customized Styles and Style List	2-9

3 Integrating Oracle Retail Applications with WebCenter Spaces

Configuring External Applications	3-1
About External Applications	3-1
Configuring Oracle Retail Applications as External Applications	3-2
Configuring Non-Compliant Applications	3-3
Adding Applications to the WebCenter Spaces Sidebar	3-6

4 Setting Up Oracle Single Sign-On

Oracle Single Sign-On Process Overview	4-2
Pre-Requisites	4-3
Configure the mod_weblogic Module	4-4
Registering the Oracle HTTP Server with the OSSO Server	4-5
Configure the mod_osso Module to Protect Web Resources	4-5
Configuring the mod_osso Module with Static Directives	4-5
Protecting URLs and Logout Dynamically (Without mod_osso)	4-7
Example: SSO Authentication with Dynamic Directives	4-7

Example: SSO Logout with Dynamic Directives.....	4-8
Add providers to your WebLogic domain for OSSO	4-8
Configure the Application for the OSSO Identity Asserter	4-10
Configuring Single Sign-On for Non-Web Retail Applications.....	4-11

Introduction

This guide describes how you can integrate Oracle Retail applications with Oracle WebCenter Spaces and customize WebCenter Spaces to include custom developed content. This document is intended for system administrators and developers, and assumes that you are familiar with the Oracle WebCenter infrastructure.

Important: In addition to this document, you must refer to the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)* available on the Oracle Technology Network. For more information on accessing this white paper, see [Accessing the Extending WebCenter Spaces White Paper](#).

Since this guide refers to the Oracle White Paper in many places, it is recommended that you review the document before you start and retain a copy for reference purposes.

This guide includes the following chapters:

- [Chapter 1, "Introduction"](#)

This chapter provides an introduction to the Oracle Retail Accelerators for WebCenter 11g and the contents of this guide.
- [Chapter 2, "Customizing WebCenter Spaces"](#)

This chapter provides an overview of the steps required to customize the WebCenter Spaces resource catalogs and page schemes. It also provides instructions on setting up a development environment before you proceed with using and customizing WebCenter Spaces.
- [Chapter 3, "Integrating Oracle Retail Applications with WebCenter Spaces"](#)

This chapter describes how you can integrate the Oracle Retail applications with WebCenter Spaces.
- [Chapter 4, "Setting Up Oracle Single Sign-On"](#)

This chapter describes how you can configure WebCenter Spaces and a select set of Oracle Retail applications to use Oracle Single Sign-On.
- [Appendix A, "References"](#)

This appendix provides a reference to relevant content in the Oracle Fusion Middleware documentation.

About Oracle Retail Accelerators

Oracle Retail Accelerators enable you to leverage the Oracle WebCenter infrastructure to create and modify workspaces specific to your business need. Apart from the capabilities that are available out-of-the-box, these enable you to create, customize, and deploy expanded functionality by combining assets from the Oracle Fusion Middleware suite.

About Oracle Retail

Oracle Retail offers the industry's most complete and integrated suite of software applications for insight-driven retailing.

About Oracle WebCenter 11g

Oracle WebCenter 11g is the industry's most complete, open, and manageable enterprise portal platform. Its unified, standards-based portal platform enables business applications to be designed by developers and evolved by business users as business requirements change.

About Oracle WebCenter Spaces

Oracle WebCenter Spaces is a ready-to-use social networking application that enables business users to quickly build individual and group work environments with a few simple clicks.

Accessing the Extending WebCenter Spaces White Paper

This section describes how you can access the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)* on the Oracle Technology Network (OTN).

You can access the white paper using the following URL:

<http://www.oracle.com/technetwork/middleware/webcenter/owcs-r11-extend-spaces-wp-132596.pdf>

OR

Navigate through the OTN Web site using the following instructions:

1. In a Web browser, log on to the Oracle Technology Network at <http://www.oracle.com/technetwork/index.html> or <http://otn.oracle.com>.
2. Under **Downloads**, in the **Middleware** section, click **Fusion Middleware 11g (incl. WebLogic)**. The **Oracle Fusion Middleware 11g Software Downloads** page appears.
3. On the left navigation pane, click **WebCenter Suite**. The **Oracle WebCenter Suite** page appears.
4. In the **Getting Started** section, click **Demonstrations and Samples**. The **Oracle WebCenter Suite 11g Demonstrations and Samples** page appears.
5. In the **Samples** section, under **WebCenter Spaces Sample**, click **Extending WebCenter Spaces**.

Customizing WebCenter Spaces

You can customize WebCenter Spaces to use custom resource catalogs and custom page schemes and styles. You can also control the specific resources and styles that will be available for use in the personal and group spaces.

This chapter provides a brief overview of the steps required to customize the WebCenter Spaces resource catalogs and page schemes. It also provides instructions on setting up a development environment before you proceed with using and customizing WebCenter Spaces. This chapter includes the following sections:

- [Setting Up a Development Environment](#)
- [Setting Up WebCenter Resource Catalogs](#)
- [Setting Up WebCenter Page Styles](#)

Important: Before you start customizing the WebCenter Spaces, in addition to this document, you must refer to the *Oracle White Paper – Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)* available on the Oracle Technology Network. For more information on accessing this white paper, see [Accessing the Extending WebCenter Spaces White Paper](#).

Setting Up a Development Environment

Before you proceed, ensure that you set up a development environment to allow customization. This section provides a summary of the infrastructure you need to set up a development environment.

For detailed information on setting up the development environment, refer to the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

Setting up a development environment involves the following:

1. Install and configure the **Oracle WebCenter Suite Release 11.1.1.2.0** or **Oracle WebCenter Suite Release 11.1.1.3.0** along with the **WebCenter Spaces** component. For more information, refer to the Oracle WebCenter documentation.
2. Download and install **Oracle JDeveloper 11g** along with **WebCenter Framework** and **Services Design Time** extensions. For more information, refer to the Oracle JDeveloper documentation.
3. Based on the Oracle WebCenter Suite release you installed, download the relevant WebCenter Spaces sample supporting file:
 - For WebCenter Suite Release 11.1.1.2.0, use the following:

<http://download.oracle.com/otndocs/tech/webcenter/files/extendwebcenterspaces.zip>

- For WebCenter Suite Release 11.1.1.3.0, use the following:

http://download.oracle.com/otndocs/tech/webcenter/files/extendwebcenterspaces_ps2.zip

4. From the supporting file you downloaded, open the customizable WebCenter Spaces workspace in JDeveloper.
5. Modify the deployment profile based on your business need.
6. Follow instructions in the Extending WebCenter Spaces white paper to build and deploy the custom WAR file.

In the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*, the section *Downloading and Modifying a Customizable WebCenter Spaces Workspaces* provides detailed information on downloading, building, and deploying the custom WAR file.

Setting Up WebCenter Resource Catalogs

This section describes how you can set up WebCenter resource catalogs and includes the following topics:

- [Resource Catalog Overview](#)
- [Setting Up a Development Environment](#)
- [Customizing the WebCenter Resource Catalogs](#)
- [Packaging and Deploying Customized Catalogs](#)

Resource Catalog Overview

The Resource Catalog provides a consolidated view of the contents of one or more otherwise unrelated repositories in a unified search and browser interface. Resources originate in their source repository and are then exposed to the authorized user through the Resource Catalog.

Resource Catalogs can contain the following components:

- **Layout components:** The primary layout component is a Box, which is a container that can hold all other types of components. At runtime, you must have a Box into which you can drag and drop components. You can also add and arrange child components and delete components from a Box.
- **Oracle ADF Faces components:** You can add Text, Image, Page Link, Web Page, and Web site Link components to your page. These are similar to the JDeveloper design time components such as Rich Text Editor, Image, Command Link, Web Page, and Go Link. The Text component enables you to add rich text on the page. Adding this component invokes a text editor that can be resized and is similar to an HTML editor. You can add text and format it using the options available in the editor.
- **Portlets:** You can add WSRP, JSF, or PDK-Java portlets from any producer that was registered in JDeveloper.
- **Task Flows:** If you have created task flows in JDeveloper (or you want to utilize the provided Oracle Retail task flows), you can add these task flows from the JAR files to your page at runtime through a customized resource catalog.

- **Documents:** If you have configured the Documents service in your application, you can add documents from this service.

Setting Up a Development Environment

To modify the WebCenter Spaces Resource Catalogs, you must first set up your development environment to allow customization of WebCenter Spaces. For more information, see [Setting Up a Development Environment](#).

Once you have set up your development environment, you can proceed ahead to customize resource catalogs.

Customizing the WebCenter Resource Catalogs

The WebCenter Spaces application provides a process to extend or customize the Resource Catalog and add new task flows, filter out content, or reorganize the folder structure. This makes it easier for your users to find the content they need.

WebCenter Spaces uses the following three resource catalogs. You can customize any one of them to fit your needs:

- **Personal space catalog** – Defines the content available to users working in their personal space. It is located at: `\oracle\adf\rc\metadata\scopedMD\defaultScope\PersonalSpaceCatalog.xml`
- **Default group space catalog** – Defines the content available to all group spaces (by default). It is located at: `\oracle\adf\rc\metadata\scopedMD\defaultScope\DefaultGroupSpaceCatalog.xml`
- **Group space catalog** – Defines the content available to a specific group space. It is located at: `\oracle\adf\rc\metadata\scopedMD\<internal_group_space_ID>\GroupSpaceCatalog.xml`

Note: The `internal_group_space_ID` is generated once a new group has been created in the WebCenter Spaces application.

To incorporate an Oracle Retail task flow, or a custom task flow that you have created, refer to the section *Customizing Resource Catalogs* in the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

Important: Any content placed in the personal space and default group space catalogs will be available to all users. Any content placed in a specific group space catalog will be available to all members of the group.

If you want to arrange catalog content so that it appears the same for all users, but displays different subsets of the content to different users, then you can configure your Resource Catalog to filter out selected items based on a specific criteria. For more information on filtering items in the Resource Catalog, refer to the section *Filtering Items in the Resource Catalog* in the *Oracle Fusion Middleware Developers Guide for Oracle WebCenter*.

Packaging and Deploying Customized Catalogs

To package and deploy the custom Resource Catalogs for use within the WebCenter Spaces application, refer to the section *Packaging and Deploying Customized Catalogs* in the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

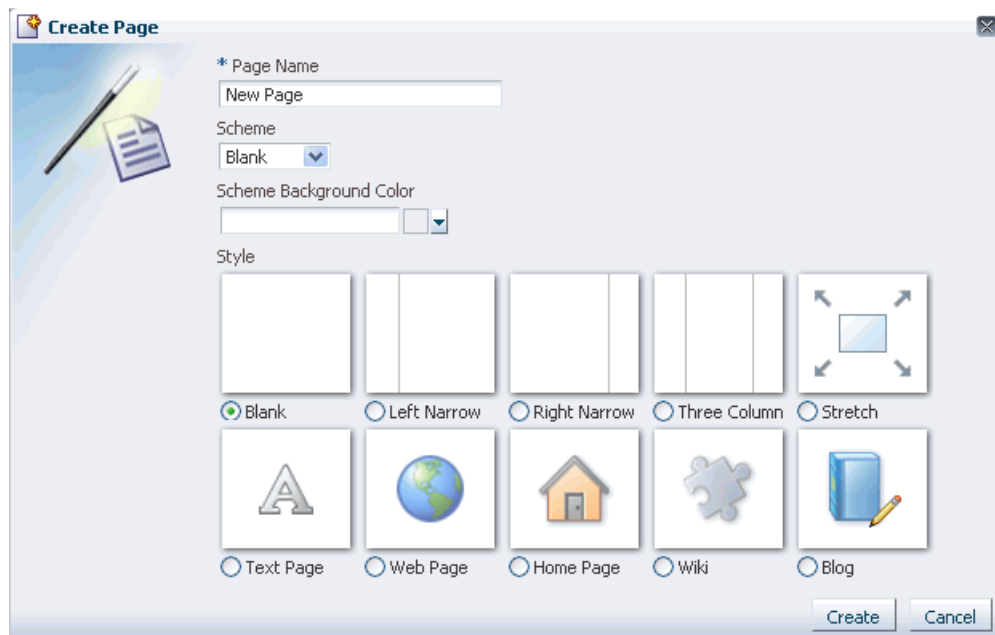
Setting Up WebCenter Page Styles

WebCenter Spaces enables authorized users to dynamically create pages at runtime in both their Personal Spaces and WebCenter Group Spaces. WebCenter Spaces provides a default set of page schemes and styles. Users can select from the schemes and styles available at runtime in the Create Page dialog box. This section highlights how you can set up WebCenter page styles and includes the following topics:

- [Setting Up a Development Environment](#)
- [Out of the Box Page Styles and Style List](#)
- [Custom Page Styles](#)
- [Creating a New Page Style](#)
- [Setting Up the Page Style List](#)
- [Packaging and Deploying the Customized Styles and Style List](#)

For more information on creating and managing pages, refer to the *Oracle Fusion Middleware User's Guide for Oracle WebCenter 11g*.

Figure 2–1 Create Page Dialog Box



On the Create Page dialog box:

- The page scheme provides seeded background images or enables you to select your own background image for the new page. The page style provides a default layout for the new page. Some page styles also automatically add ADF components or taskflows to the page that can be further customized by users at runtime.

- You can customize WebCenter Spaces to provide custom page schemes and styles. You can also control the styles that are available for use in personal or group spaces. For more information, refer to the section *Modifying Page Styles* section in the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

Setting Up a Development Environment

To modify the WebCenter Spaces Resource Catalogs, you must first set up your development environment to allow customization of WebCenter Spaces. For more information, see [Setting Up a Development Environment](#).

Once you have set up your development environment, you can proceed ahead to customize resource catalogs.

Out of the Box Page Styles and Style List

The Extend WebCenter Spaces ZIP file includes the out-of-the-box page styles and page style list provided by WebCenter spaces in the following locations:

- `<Unzip_Dir>\ExtendWebCenterSpaces\SourceFiles\page_styles`
- `<Unzip_Dir>\ExtendWebCenterSpaces\SourceFiles\page_styles\oracle\webcenter\page\templates\templates.xml`

You may refer to the out-of-the-box styles as examples of properly constructed page style templates.

Custom Page Styles

To override the out-of-the-box styles, you must modify the CustomPageStyle project and war. Sample custom page styles are included in the CustomPageStyle project at `<Unzip_Dir>\ExtendWebCenterSpaces\CustomPageStyle\public_html\custom\oracle\webcenter\page\templates`

Creating a New Page Style

A page style template is a JSPX page along with an associated page definition file. It defines the layout areas for pages created based on the style template. The page style templates can also include task flows, text, images, and ADF components to standardize the look and feel of the page or to give the users a jump-start when creating a new page. The page template gets copied when a new page is created based off of the template.

To create a new page style template:

1. Create a `<page style>.jspx` page (where `<page style>` is the name of your new page style template), or customize one of the sample page style JSPX pages in the **CustomPageStyle** project.

When creating your own JSPX page, review one of the existing pages in the **CustomPageStyle** project for an example. The **TemplateBlank.jsx** file is a good example. The first thing you will notice in the sample page style templates is the value for the document title, similar to the following:

```
<af:document title="#{pageDocBean.title}" id="docrt">
```

At run time the title is set based on the value of the WebCenter Spaces pageDocBean's title attribute. You must specify the document title in the same way in your custom page style template.

You will also notice that page style templates generally are based on WebCenter Spaces' current site template, similar to the following:

```
<af:pageTemplate
viewId="#{WCAppContext.application.siteTemplatesManager.currentSiteTemplateView
Id}" value="#{bindings.shellTemplateBinding}" id="T">
```

The page template has a "content" facet in which you will include the tags that will lay out the content of your page. The following example illustrates how the blank content is set up in the TemplateBlank.jspx file:

```
<f:facet name="content">
  <pe:pageCustomizable id="pcl1">
    <af:panelStretchLayout id="psl2"
      styleClass="replace_with_scheme_name"
      inlineStyle="replace_with_inline_style">
      <f:facet name="center">
        <af:panelGroupLayout id="pgl1"
          layout="scroll">
          <pe:layoutCustomizable id="lc1"
            showLayoutChanger="#{pageServiceBean.isEditMode}"
            text="#{uib_o_w_w_r_WebCenter.LABEL_CHANGE_LAYOUT}"
            showIcon="false" type="oneColumn"
            shortDesc="#{uib_o_w_w_r_WebCenter.LABEL_CHANGE_LAYOUT}">
            <cust:panelCustomizable id="mainC"/>
            <f:facet name="contentA">
              <cust:panelCustomizable id="cnta"/>
            </f:facet>
            <f:facet name="contentB">
              <cust:panelCustomizable id="cntb"/>
            </f:facet>
          </pe:layoutCustomizable>
        </af:panelGroupLayout>
      </f:facet>
    </af:panelStretchLayout>
    <f:facet name="editor">
      <pe:pageEditorPanel id="pep1"/>
    </f:facet>
  </pe:pageCustomizable>
</f:facet>
```

The content facet in the **TemplateBlank.jspx** file starts with a `<pe:pageCustomizable>` tag. If you want new pages to be editable at run time, you must include the content within a Page Customizable component.

Within `<pe:pageCustomizable>`, under the `<af:panelStretchLayout>` tag, note the values of the `styleClass` and `inlineStyle` attributes:

```
styleClass="replace_with_scheme_name"
inlineStyle="replace_with_inline_style">
```

If you create your own page style templates and want to set a page style, you must have a Panel Group Layout or Panel Stretch Layout component as a direct child of the Page Customizable. Also, you must set the placeholder strings for `styleClass` and `inlineStyle`, as shown in the example above. When WebCenter Spaces creates the page from your template, the `replace_with_scheme_name` string is replaced with the value the users selected in the **Scheme** field and the `replace_with_inline_style` string is replaced with the value selected in the **Scheme Background** field. The **Scheme** and **Scheme Background** fields appear on the **Create Page** dialog box.

The **TemplateBlank.jspx** file also uses the `<pe:layoutCustomizable>` tag. The Layout Customizable component supports changing the layout at run time. Review the value of the `showLayoutChanger` attribute in the following example:

```
showLayoutChanger="#{pageServiceBean.isEditMode}"
```

If the value of the `showLayoutChanger` attribute of any Layout Customizable component is set to `#{pageServiceBean.isEditMode}`, the "layout changer" icon will appear only when the user is editing the page.

For more information on creating page style templates, refer to the *Oracle Fusion Middleware Developer's Guide for WebCenter 11g Release 1 (Section 6.6 – Introduction to Custom Styles and Templates in the Chapter 6 Enabling Runtime Creation and Management of Pages)*.

2. Create an associated **<page style>PageDef.xml** file. Every page style template requires a corresponding Page Definition (PageDef) XML file. For example, the **TemplateBlank.jspx** file has a corresponding **TemplateBlankPageDef.xml** with the following content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
  version="11.1.1.41.30"
  id="ps_pagedefusage"
  Package="ps_package">
  <parameters/>
  <executables>
    <page path="oracle.webcenter.webcenterapp.bindings.pageDefs.
      oracle_webcenter_webcenterapp_view_templates_
      WebCenterAppShellTemplatePageDef"
      id="shellTemplateBinding" Refresh="ifNeeded"/>
    <taskFlow id="pageeditorpanel"
      taskFlowId="#{pageEditorBean.pageEditorPanel}"
      xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
  </executables>
  <!-- Page Permission mapping -->
  <permission permissionClass="oracle.webcenter.page.model.security.
    CustomPagePermission" target="ps_targetusage"
    xmlns="http://xmlns.oracle.com/adf/security">
    <privilege-map operation="administer" privilege="manage"/>
    <privilege-map operation="create" privilege="create"/>
    <privilege-map operation="delete" privilege="delete"/>
    <privilege-map operation="edit" privilege="update"/>
    <privilege-map operation="personalize" privilege="personalize"/>
    <privilege-map operation="view" privilege="view"/>
  </permission>
</pageDefinition>
```

Note the following in the **TemplateBlankPageDef.xml** example:

- The values of the `id` and `Package` attributes in the `pageDefinition` element. This syntax must be followed in your page style template page definition XML file. The `ps_pagedefusage` and `ps_package` strings are replaced at run time with the relevant values for the newly created page.
- The `<page>` element, within the `<executables>` element, binds to the `pageTemplate` of the JSPX page.
- The `<taskFlow>` element has an `ID` attribute with a value `pageeditorpanel` and a `taskFlowId` attribute with a value `#{pageEditorBean.pageEditorPanel}`. This task flow corresponds to

the editor facet of the Page Customizable component in the **TemplateBlank.jspx** file.

- The `<permission>` element has a `target` attribute with a value `ps_targetusage`. The value for `ps_targetusage` gets substituted at run time when the page is created.
3. Copy the **<page style>.jspx** and **<page style>PageDef.xml** files to `\custom\oracle\webcenter\page\templates`.
 4. Edit the page style list (located in `\custom\oracle\webcenter\page\templates\templates.xml`) to include your new page style template. For more information, see [Setting Up the Page Style List](#).

For additional information, refer to section *Creating a New Page Style* in the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

Setting Up the Page Style List

The page style list determines the page style selections available in the Create Page dialog box. When you create new page style templates, you must add them to the page style list to make them visible in the Create Page dialog box. It is also possible to edit the page style list to change the order of style templates, designate the styles that apply to Group Spaces, and delete page style templates from the list.

When extending WebCenter Spaces, the page style list is defined in the `templates.xml` file located at `\custom\oracle\webcenter\page\templates\` in the CustomPageStyle project.

To set up the page styles list:

1. Make an editable copy of the **templates.xml** file.
2. Add, edit, rearrange, or delete the `<templateDef>` elements in your copy of the **templates.xml** file.

For example,

To add a **Group Space** style for a page style named **GroupSpaceTemplateDashboard.jspx**, insert a `<templateDef>` element similar to the following:

```
<templateDef name="GroupSpaceTemplateDashboard.jspx" title="Dashboard"
icon="/custom/oracle/webcenter/page/templates/images/gsDashboard.png"
forGroupSpace="true"/>
```

To add a page style named **TemplateDashboard.jspx**, since this template is not for group spaces, insert a `<templateDef>` element similar to the following:

```
<templateDef name="TemplateDashboard.jspx" title="Dashboard"
icon="/custom/oracle/webcenter/page/templates/images/dashboard.png"
forGroupSpace="false"/>
```

For more information on the syntax for `<templateDef>` elements, refer to the *Oracle Fusion Middleware Developer's Guide for WebCenter 11g Release 1 (Section 6.3.4.1 – Out-of-the-Box Styles in the Chapter 6 – Enabling Runtime Creation and Management of Pages)*.

3. Copy the modified **templates.xml** back to the `\custom\oracle\webcenter\page\templates` directory.

Packaging and Deploying the Customized Styles and Style List

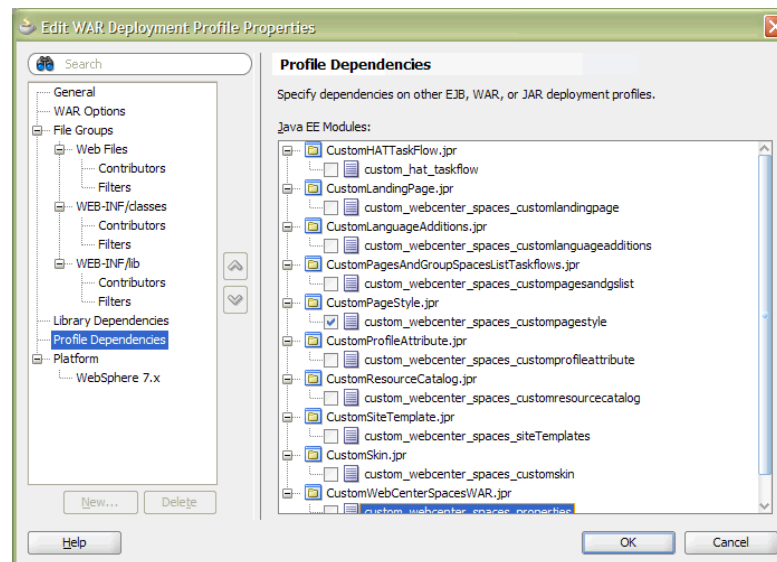
Once you have completed your new page style templates and modified the style list, you must package the page style templates and style list into a JAR file for deployment. Once the JAR file is ready, you must add it to the WebCenter customization shared library, and then deploy the customizations to WebCenter.

This section provides an overview of the steps you must complete to package and deploy the customized page styles. For more information on packaging and deploying the customized page styles, refer to the instructions in the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

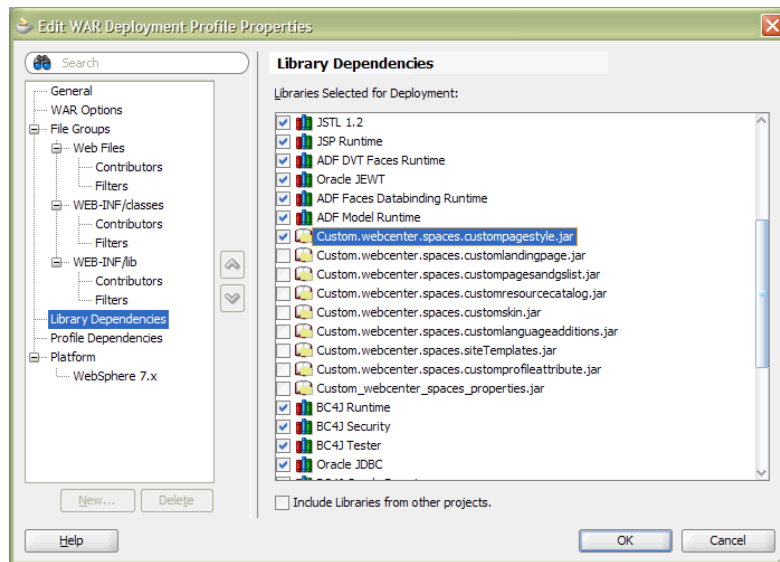
To package and deploy the customized styles and styles list:

1. Copy the **templates.xml** file to
`\custom\oracle\webcenter\page\templates` directory, if you have not already done so.
2. Copy any images to the relevant directories. For example,
`\custom\oracle\webcenter\page\templates\images`.
3. In **Oracle JDeveloper**, select the **CustomPageStyle** project for the **CustomWebCenterSpacesWAR** project in the **Profile Dependencies** section of the **Edit WAR Deployment Profile Properties** window.

Figure 2–2 Edit WAR Deployment Profile Properties Window – Profile Dependencies



4. In the **Library Dependencies** section of the **Edit WAR Deployment Profile Properties** window, ensure that the Custom Page Style JAR file is included for the **CustomWebCenterSpacesWAR** project.

Figure 2–3 Edit WAR Deployment Profile Properties Window – Library Dependencies

5. Deploy the custom page styles by deploying the Custom WebCenter Spaces shared library.

For more information on deploying the custom WebCenter Spaces shared library, refer to the Oracle White Paper – *Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0)*.

Integrating Oracle Retail Applications with WebCenter Spaces

The Applications pane in the WebCenter Spaces Sidebar offers users quick access to the applications they use most. It can display links to external applications registered through the Fusion Middleware Control Console and links to built-in WebCenter task flows. The WebCenter Spaces administrator manages the content of the Applications pane, including the available applications, the way they are presented, and how they are launched. Users can personalize the Applications pane to hide the links to applications they do not use. Users can also personalize the WebCenter Spaces Favorites menu to add links to favorite web pages, including Oracle Retail applications.

This chapter describes how you can integrate Oracle Retail applications in WebCenter Spaces. It includes the following sections:

- [Configuring External Applications](#) – provides information on registering the applications as external applications using Fusion Middleware Control.
- [Adding Applications to the WebCenter Spaces Sidebar](#) – provides information on adding the applications to the WebCenter Spaces Application pane.

Configuring External Applications

This section provides an overview of external applications and highlights the steps you must complete to configure external applications. It includes the following topics:

- [About External Applications](#)
- [Configuring Oracle Retail Applications as External Applications](#)
- [Configuring Non-Compliant Applications](#)

About External Applications

Oracle WebCenter Framework defines an external application as any application that implements its own authentication process. That is, an application that does not take part in a WebCenter application's single sign-on process. In some cases, the identity management solution may be the same, but the authentication process can be different.

When WebCenter Framework Service interacts with an application that handles its own authentication, you can associate that service with an external application to allow for credential provisioning. Therefore, the use of an external application definition provides a means of accessing content from these independently authenticated applications.

To replicate a single sign-on experience from the end user's perspective, the external application service captures the user name and password, along with any other credentials for the external application, and supplies it to the WebCenter service requiring it. The WebCenter service then uses this and logs in on behalf of the end user. This user name and password combination is securely stored in a credential store configured for the WebLogic domain where the application is deployed.

The user provides the login credentials when prompted, and these credentials are mapped to the WebCenter application user and stored in the credential store configured for the domain. The credential store subsequently supplies that information during authentication to the external application. Unless the external application's credentials change, the user supplies the credentials only once as the mapped information is read from the credential store for future requests.

Configuring Oracle Retail Applications as External Applications

To add Oracle Retail Web-based applications to the WebCenter Spaces Sidebar, you must first register them as external applications using Fusion Middleware Control or WLST commands.

For more information on registering such external applications, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter (Section 22.2 – Registering External Applications in the Chapter 22 – Managing External Applications)*.

Follow the instructions that are specific to WebCenter Spaces. You will be required to register information about the application, including the URL of the login page, the names of the application's user ID and password fields, the application's authentication method, and any optional parameters included in the login page. The document also explains how you can determine the values by examining the HTML source of the application's login form.

[Table 3–1](#) lists the external application registration parameters for the Web-based Oracle Retail applications. If you want to configure the Oracle Retail Web-based applications as external applications with automated login support, ensure that they are not configured for SSO.

Once you have completed the external application registration, you can add the applications to the WebCenter Spaces Sidebar. For more information, see [Adding Applications to the WebCenter Spaces Sidebar](#).

Table 3–1 External Application Registration Parameters for Web-based Oracle Retail Applications

Application Name	Registration Information	Value
Allocation	Login URL	http://<hostname>:<port>/<context>/gencookie.jsp?action=login where <i>hostname</i> , <i>port</i> and <i>context</i> refer to where the Allocation application is deployed.
	HTML User ID Field Name	user
	HTML User Password Field Name	password
	Authentication Method	POST

Table 3–1 (Cont.) External Application Registration Parameters for Web-based Oracle Retail Applications

Application Name	Registration Information	Value		
Invoice Matching	Login URL	The full URL to the Invoice Match application; get it from the application administrator		
	HTML User ID Field Name	username		
	HTML User Password Field Name	password		
	Authentication Method	POST		
Advanced Inventory Planning (AIP)	Login URL	http://<hostname>:<port>/<context>/servlet/rfpservlet/trlogin where <i>hostname</i> , <i>port</i> and <i>context</i> refer to the location where the AIP application is deployed.		
	HTML User ID Field Name	username		
	HTML User Password Field Name	password		
	Authentication Method	POST		
	Additional Login Fields	Name	Value	Display to user
		enterpriseid	aiponline	leave unchecked

For more information on external applications, refer to the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter (Section 37.2 – Working with External Applications in the Chapter 37 – Securing Your WebCenter Application)*.

Configuring Non-Compliant Applications

Some of the Oracle Retail applications cannot be configured as true external applications with automated login support. External applications must be Web-based applications and handle their own authentication. In order to fully support automated login, an application must:

- Include User ID and password fields on the application's login page.
- Support post or get authentication methods.
- Support UTF8 encoding.
- Not be configured for SSO.

Note: Applications with a customized login page built using ADF Faces must implement the J2EE security container login method `j_security_check` for authentication.

Implement the J2EE security container login method `j_security_check` for authentication. This applies to applications with a customized login page (built using ADF Faces).

In this document, the Oracle Retail applications that cannot be configured as external applications with automated login support are termed as Non-Compliant applications. Such applications can be classified in the following categories:

- Applications that have been SSO enabled.

- Java WebStart applications, such as RPM and SIM.
- Oracle Forms applications, such as RMS, RWMS, and ARI.

These Oracle Retail applications cannot officially be configured as external applications with automated login and are not officially supported as external applications. However, it may be possible to work around these constraints, configure them as external applications, and include them in the WebCenter Spaces Sidebar.

Note the following limitations:

- Although the values will be ignored, you must provide random values for the HTML User ID and HTML User Password field names. Ensure that the field name values do not conflict with query string parameters supported by the application.
- You must configure the applications with the GET authentication method.
- Certain applications will require configuration of additional login fields.
- When the users access the application for the first time, the External Applications login screen will appear. Users can enter any values for user name and password, as these values will be ignored.

To avoid the initial External Applications login screen, users can choose to set up the random user name and password values in the My Accounts tab of the WebCenter Spaces Preferences dialog box.

- Single sign-on is not possible for applications that are not SSO-enabled because these applications present their own login screen when accessed.

Note: Users logged in to WebCenter through OSSO can directly access the SSO-enabled applications that are configured in the same OSSO infrastructure without having to enter the user credentials again.

[Table 3–2](#) summarizes the work around for Non-Compliant applications. Once you have registered such applications, you can add the applications to the WebCenter Spaces Sidebar. For more information, see [Adding Applications to the WebCenter Spaces Sidebar](#).

Table 3–2 Work Around Parameters for Non-Compliant Oracle Retail Applications (unsupported)

Application Name	Registration Information	Value		
RPM, SIM, OSSO-enabled applications	Login URL	Specify the URL to the application, excluding query string parameters such as the template parameter. For example: <code>http://<hostname>:<port>/Jnlplaunch/launch?</code> where <i>hostname</i> and <i>port</i> refer to the location where the application is deployed.		
	HTML User ID Field Name	Any string that does not conflict with query string parameter names that are supported by the application. It is recommended that you use a string which indicates the value is ignored. For example, "username-ignored".		
	HTML User Password Field Name	Any string that does not conflict with query string parameter names that are supported by the application. It is recommended that you use a string which indicates the value is ignored. For example, "password-ignored".		
	Authentication Method	GET		
	Additional Login Fields for RPM and SIM Note: Ensure that any other required query string parameters are also added as additional login fields.	Name	Value	Display to user
		template	The name of the application's jnlp template. For example, "rpm_jnlp_template.vm" or "sim_jnlp_template.vm".	leave unchecked
Forms-based applications	Login URL	Specify the URL to the forms servlets, excluding any query string parameters. For example: <code>http://<hostname>:<port>/forms/frmservlet?</code> where <i>hostname</i> and <i>port</i> refer to the location where the application is deployed.		
	HTML User ID Field Name	Any string that does not conflict with query string parameter names that are supported by the application. It is recommended that you use a string which indicates the value is ignored. For example, "username-ignored".		
	HTML User Password Field Name	Any string that does not conflict with query string parameter names that are supported by the application. It is recommended that you use a string which indicates the value is ignored. For example, "password-ignored".		
	Authentication Method	GET		
	Additional Login Fields	Name	Value	Display to user
		config	The value of the Forms application's configuration parameter.	leave unchecked

Adding Applications to the WebCenter Spaces Sidebar

Once you have registered the Oracle Retail applications as external applications, you can customize the WebCenter Spaces Sidebar to display links to those applications in the Applications pane. This section describes how you can make an Oracle Retail application appear in the WebCenter Spaces Sidebar.

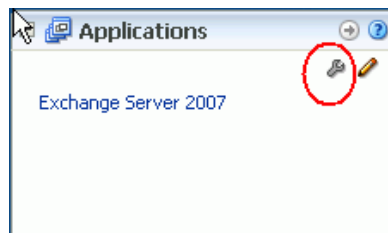
Note: When you make the applications appear in the Applications pane, the applications become available to all WebCenter users.

Some users may not want to see all of the applications, or may not have relevant access to some of the applications in the Applications pane. Users can choose to personalize their view and list only those applications they want to access.

To add an Oracle Retail application to the WebCenter Spaces Sidebar:

1. Log on to WebCenter Spaces with administrative privileges.
2. In the Sidebar, click the Application pane's **Edit** icon.

Figure 3–1 Edit Icon in the Application Pane



3. To add a link to an application, select the folder where you want to add the application, then click the **Add** icon.
4. Navigate to the application you want to add and click its associated **Add** link.

If you want to add new folders, click the **New** icon.

For more information on the making an application to WebCenter users, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter 11g* (Section 36.2 – Making an Application Available to WebCenter Users in the Chapter 36 – Making Applications Available in WebCenter Spaces).

Setting Up Oracle Single Sign-On

Oracle Single Sign-On (OSSO) provides single sign-on access for Web-based applications. This chapter provides an overview of OSSO and describes how you can configure a system to use OSSO with WebCenter. It contains the following sections:

- [Oracle Single Sign-On Process Overview](#)
- [Pre-Requisites](#)
- [Configure the mod_weblogic Module](#)
- [Registering the Oracle HTTP Server with the OSSO Server](#)
- [Configure the mod_osso Module to Protect Web Resources](#)
- [Add providers to your WebLogic domain for OSSO](#)
- [Configure the Application for the OSSO Identity Asserter](#)
- [Configuring Single Sign-On for Non-Web Retail Applications](#)

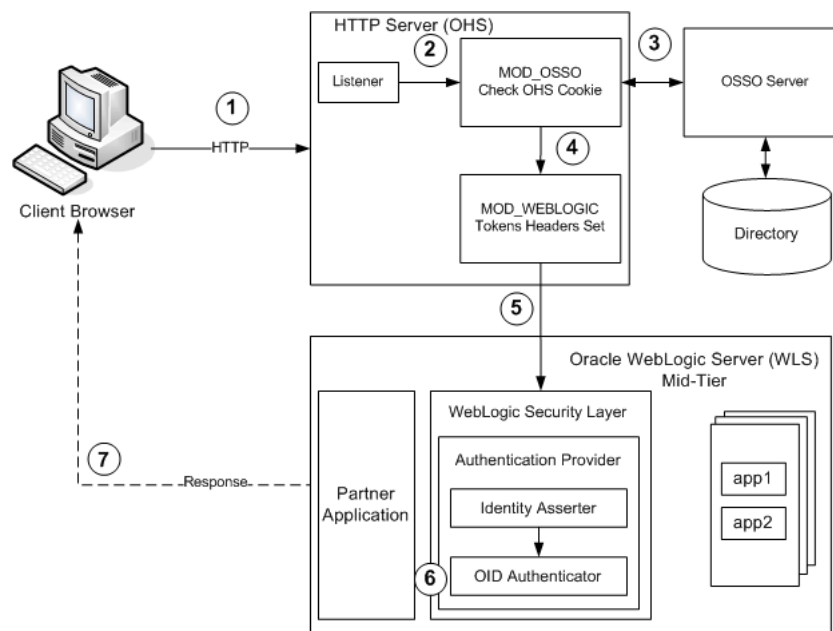
The configuration described in this chapter is intended for applications that have been deployed on Oracle WebLogic Server but do not yet have single sign-on implemented. OSSO requires an Oracle Internet Directory server, an LDAP repository storing an enterprise's users and groups. Integration with other LDAP servers may be performed through an Oracle Virtual Directory server.

Note: OSSO-enabled applications cannot be configured as true external applications. However, it may be possible to work around these constraints, configure them as external applications, and include them in the WebCenter Spaces Sidebar. For more information, see [Configuring Non-Compliant Applications](#) and [Adding Applications to the WebCenter Spaces Sidebar](#).

Oracle Single Sign-On Process Overview

The `osso_module` within the Oracle HTTP Server checks each incoming request to a protected URL to see if the current user requires the authentication process to begin. Once a user is authenticated through OSSO, the `mod_osso` module creates secure headers used by OID Authenticator in the WebLogic Server to provide the application with authenticated user's identity. A typical process overview is illustrated in the following diagram:

Figure 4–1 Oracle Single Sign-On Process Overview



1. The user's Web browser makes an HTTP request to a protected URL serviced by the Oracle HTTP Server (OHS).
2. The Oracle HTTP Server processes the request and routes it to the `mod_osso` module.
3. This module determines whether the user is already authenticated. If authentication is required, it directs the browser to the OSSO server. The OSSO server checks for a secure cookie containing the authentication information. If the cookie is not found, the following occurs:
 - a. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
 - b. 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. The browser is then redirected back to the Oracle HTTP Server with the encrypted authentication credentials. It does NOT contain the user's password.
4. The `mod_osso` module then decrypts the user credentials and sets HTTP headers with relevant user attributes, marking the user's session as authenticated.
5. The `mod_weblogic` module (within the Oracle HTTP Server) then forwards the request to the Oracle WebLogic Server.

6. The Oracle WebLogic Server then invokes the configured authentication providers that decode the headers and provide the user's role membership. In an OSSO implementation, ensure that the OSSO Identity Asserter is invoked and Oracle Internet Directory (OID) Authenticator is executed to provide the user's role membership.
7. Once the authentication is established, the relevant application logic is initiated and the response is sent back to the user through the Oracle HTTP Server.

Because the Web browser session is now authenticated, subsequent requests in that session are not redirected to the OSSO server for authentication.

Pre-Requisites

In a default installation, WebCenter uses the HTTP server provided by the WebLogic Server. To use WebCenter with Oracle Single Sign-On, an Oracle HTTP Server must be set up as the front-end to the WebLogic Server. OSSO specific logic is implemented within the Oracle HTTP Server by the `mod_osso` module. The Oracle HTTP Server is provided as part of the Oracle WebTier package. Additional configuration is also needed for the WebLogic Server hosting WebCenter.

Before proceeding, ensure that you have the following infrastructure installed:

- Oracle Application Server Single Sign-On (OSSO). For more information, refer to the *Oracle Application Server Installation Guide* included within the *Oracle Identity Management 10g Release 3 (10.1.4)* documentation.
- Oracle Identity Management Infrastructure server, including the Oracle Internet Directory (OID) LDAP. For more information, refer to the chapter *Installing OracleAS Infrastructure* in the *Oracle Application Server Installation Guide 10g (10.1.4.0.1)*.
- Oracle HTTP Server 11g as a front end to the Oracle WebLogic Server. For more information, refer to the *Oracle Fusion Middleware Installation Guide for Oracle Web Tier 11g Release 1 (11.1.1)*.
- Oracle WebLogic Server 11g Release 1 (10.3.2). For more information, refer to the Oracle WebLogic Server documentation.
- Oracle Fusion Middleware product such as Oracle Identity Management, Oracle SOA Suite, or Oracle WebCenter. This package includes the provider required for OSSO by Oracle WebLogic Server at the following location:

`ORACLE_INSTANCE/modules/oracle.ossoiap_11.1.1/ossoiap.jar`

For more information, refer to the *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*, *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite*, or *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

Configure the mod_weblogic Module

The Oracle HTTP Server uses the `httpd.conf` file as its base configuration file. Ensure that the `httpd.conf` includes entries for the `mod_weblogic` module or references the `mod_weblogic` module configuration file (`mod_wl_ohs.conf`) using the *include* directive. Additional configuration files can be referenced using an *include* directive.

Configure the `mod_weblogic` module using the following steps:

1. Navigate to the location where the **`httpd.conf`** file for the Oracle HTTP Server exists and open it for editing. For example,

```
${WT_ORACLE_HOME}/instances/<WebTier_Instance>/config/OHS/<ohs_id>/httpd.conf
```

Where,

- `${WT_ORACLE_HOME}` is the base directory where the Oracle HTTP Server is installed.
- `<WebTier_Instance>` is the name of the Oracle WebTier instance.
- `<ohs_id>` is the name assigned to the Oracle HTTP Server that acts as the front end to the Oracle WebLogic Server. For example, `ohs1`, `ohs2`, and so on.

2. Verify that the reference to the **`mod_weblogic`** configuration file (**`mod_wl_ohs.conf`**) exists in the **`httpd.conf`** file. For example,

```
${WT_ORACLE_HOME}/instances/<WebTier_Instance>/config/OHS/<ohs_id>/mod_wl_ohs.conf
```

Where,

- `${WT_ORACLE_HOME}` is the base directory where the Oracle HTTP Server is installed.
- `<WebTier_Instance>` is the name of the Oracle WebTier instance.
- `<ohs_id>` is the name assigned to the Oracle HTTP Server that acts as the front end to the Oracle WebLogic Server. For example, `ohs1`, `ohs2`, and so on.

The `mod_wl_ohs.conf` includes content similar to the following:

```
LoadModule weblogic_module "${ORACLE_HOME}/ohs/modules/mod_wl_ohs.so"
#This empty block is needed to save mod_wl related configuration from EM to
this file when changes are made at the Base Virtual Host Level
<IfModule weblogic_module>
    WebLogicHost <host name>
    WebLogicPort <admin port>
    WLLogFile /tmp/mod_wl_ohs.log
    MatchExpression *.jspx
</IfModule>

<Location /<context-root>
    SetHandler weblogic-handler
    WebLogicHost <host name>
    WebLogicPort <port>
</Location>
```

Registering the Oracle HTTP Server with the OSSO Server

You must register the Oracle HTTP Server's host name and TCP port number with the OSSO server using the `ssoreg.sh` script (`ssoreg.bat` on Windows-based system). The output of this command will be a binary file (`osso.conf`). Copy the `osso.conf` file to the Oracle HTTP server and reference it in the `mod_osso` module. For more information, refer to the following documentation:

- *Oracle Application Server Single Sign-On Administrator's Guide 10g Release 3 (10.1.4).*
- *Oracle Identity Management Application Developer's Guide 10g Release 3 (10.1.4).*

The following procedure illustrates a sample command to register `mod_osso`:

1. Navigate to the following location in the Oracle Identity Management directory:

```
ORACLE_HOME/sso/bin/ssoreg
```

2. Run the `ssoreg` script with the following parameters. For example, on a Unix-based system:

```
./ssoreg.sh -oracle_home_path ${ORACLE_HOME} -site_name <wls_server >
-config_mod_osso TRUE -mod_osso_url <OHS URL>
-update_mode CREATE -remote_midtier -config_file <output_osso_conf_file>
```

Where,

- `${ORACLE_HOME}` is the `ORACLE_HOME` directory for the SSO server installation.
- `<wls_server>` is the name of the OHS site. Typically, this is the name of the server's host computer.
- `<OHS_URL>` is the URL used to access the Oracle HTTP Server. For example, `http://sampleOHS.com:7778`
- `<output_osso_conf_file>` is the name or location of the output `osso.conf` file. Ensure that this file is copied to the Oracle HTTP Server instance and referenced within the `mod_osso.conf` configuration file.

Configure the mod_osso Module to Protect Web Resources

The `mod_osso` module redirects the user to the single sign-on server only when the URL requested is configured to be protected. You can protect the URLs in one of the following ways:

Static protection is performed using the configuration entries in the `mod_osso.conf` file.

Dynamic protection is performed by an application returning a specific response error code to force the user to be authenticated. Dynamic protection enables a user to access a URL in a non-privileged manner first, and then see any privileged data once logged in.

Configuring the mod_osso Module with Static Directives

You can statically protect URLs with the `mod_osso` module by applying directives to the `mod_osso.conf` file. These configuration entries contain the location of the protected URIs, time out interval, and the authentication method.

Note: In the httpd.conf file, it is recommended that you place the include statement for the mod_osso.conf file before the weblogic_module statement entry.

To configure the mod_osso module to protect the Web resources:

1. Copy the **osso.conf** file from the location where it was generated to the following location:

```
${WT_ORACLE_HOME}/instances/<WebTier_Instance>/config/OHS/<ohs_id>/osso/osso.conf
```

Where,

- `${WT_ORACLE_HOME}` is the base directory where the Oracle HTTP Server is installed.
 - `<WebTier_Instance>` is the name of the Oracle WebTier instance.
 - `<ohs_id>` is the name assigned to the Oracle HTTP Server that acts as the front end to the Oracle WebLogic Server. For example, ohs1, ohs2, and so on.
2. Copy the **mod_osso.conf** file from the disabled directory to the **moduleconf** directory for editing. For example:

```
cd ${WT_ORACLE_HOME}/instances/<WebTier_Instance>/config/OHS/<ohs_id>  
cp ./disabled/mod_osso.conf ./moduleconf/mod_osso.conf
```

3. Edit the **mod_osso.conf** file and add the following information using values for your deployment.

For example, the mod_osso.conf file protecting the initial context root "root" and all URIs contains the following:

```
LoadModule osso_module "${ORACLE_HOME}/ohs/modules/mod_osso.so"  
<IfModule osso_module>  
  OssoIpCheck on  
  OssoIdleTimeout off  
  OssoConfigFile ORACLE_INSTANCE/config/OHS/<ohs_name>/osso/osso.conf  
  #Location is the URI you want to protect  
  <Location /root>  
    require valid-user  
    AuthType Osso  
  </Location>  
</IfModule>
```

Note: You can specify multiple locations by specifying multiple `<Location>` elements.

4. Navigate to the following location:

```
${WT_ORACLE_HOME}/instances/<WebTier_Instance>/config/OHS/<ohs_id>/
```

5. Edit the **httpd.conf** file and confirm that the mod_osso.conf file path for your environment is included. For example:

```
include moduleconf/mod_osso.conf
```

6. Restart the Oracle HTTP Server.

Protecting URLs and Logout Dynamically (Without mod_osso)

Applications that use dynamic directives require no entry in the mod_osso.conf file because mod_osso protection is implemented via the application logic.

Dynamic directives are HTTP response headers having special error codes that enable an application to request granular functionality from the single sign-on system without having to implement the specifics of the single sign-on protocol. Upon receiving a directive as part of a simple HTTP response from the application, the mod_osso module creates the relevant single sign-on protocol message and communicates it to the single sign-on server.

The Oracle Application Server supports the dynamic directives for Java servlets and JSP files. The product does not currently support dynamic directives for PL/SQL applications. The JSP files that follow show how such directives are incorporated. Similar to their "static" counterparts, these sample "dynamic" applications generate user information.

Example: SSO Authentication with Dynamic Directives

The home.jsp file includes a ssodynauth.jsp that uses the request.getUserPrincipal().getName() method to check the user in the session. If the user is absent, it issues a dynamic directive 499, a request for simple authentication. In the following code sample, key lines are highlighted in bold:

```
//home.jsp

<%@ include file="ssodynauth.jsp" %>
<%
//page content goes here
%>

//ssodynauth.jsp

<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "0");
%>
<%
// Check for user
String ssoUser = null;
try
(
//ssoUser = request.getRemoteUser();
ssoUser = request.getUserPrincipal().getName( );
ssoUser = ssoUser.trim( );
}
catch(Exception e)
{
ssoUser = null;
}

// If user is not authenticated then generate
// dynamic directive for authentication
if((ssoUser == null) || (ssoUser.length() < 1))
{
response.sendError(499, "Oracle SSO");
return;
}%>
```

Additional Documentation

For more information, refer to the *Oracle Identity Management Application Developer's Guide 10g (10.1.4.0.1)*.

Example: SSO Logout with Dynamic Directives

To achieve a global logout, also known as single log-out, applications are expected to first invalidate sessions and then make a call to OSSO logout. The logout.jsp file issues a dynamic directive 470, a request for OSSO logout. The osso-return-logout is set by the application to specify the return URL after logout.

In Oracle Fusion Middleware 11g, the SSOFilter handles sessions synchronization. In the following example, the key lines for SSO logout with dynamic directives appear highlighted in bold:

```
//logout.jsp
<%@page session="false"%>
<%
    response.setHeader("Osso-Return-Url", "http://my.oracle.com/");
    HttpSession session = null;
    session = request.getSession();
    if (null != session )
    {
        // necessary for achieving SLO
        session.invalidate();
    }
    response.sendError(470, "Oracle SSO");
%>
```

Additional Documentation

For more information, refer to the *Oracle Identity Management Application Developer's Guide 10g (10.1.4.0.1)*.

Add providers to your WebLogic domain for OSSO

You must add the OSSO Identity Asserter to a WebLogic domain. In addition to the OSSO Identity Asserter, Oracle recommends the following Authentication providers:

- DefaultAuthenticator
- OID Authenticator

To add providers to your WebLogic domain for OSSO Identity Assertion:

1. Log on to the WebLogic Administration Console.
2. Under the **Domain Structure** (left navigation pane), click **Security Realms**. The **Summary of Security Realms** screen appears.
3. On the **Summary of Security Realms** screen, click the default security realm (myrealm). The **Settings for myrealm** screen appears.
4. On the **Settings for myrealm** screen, click the **Providers** tab, and then click **New**. The **Create a New Authentication Provider** screen appears.
5. Enter a provider name for the OSSO Identity Asserter, select the relevant type, and then click **OK**. For example:

Name: OSSO Identity Asserter

Type: OSSOIdentityAsserter

The new provider is added to the list of providers and appears on the **Settings for myrealm** screen.

6. Click the name of the provider you just added.
7. On the **Common** tab, set the relevant values for the parameter, set the **Control Flag** value to **Sufficient**, and then click **Save**.
8. On the **Providers** tab, click **DefaultAuthenticator**. The **Settings for DefaultAuthenticator** screen appears.
9. Set the **Control Flag** value to **Optional** and click **Save**.
10. On the **Providers** tab, click **New**. The **Create a New Authentication Provider** screen appears.
11. Enter a provider name for the **OID Authenticator**, select the relevant type, and then click **OK**. For example:

Name: OID Authenticator

Type: OracleInternetDirectoryAuthenticator

The new provider is added to the list of providers and appears on the **Settings for myrealm** screen.

12. Click the name of the provider you just added and review the settings. Do not change the Control Flag value until you have verified that the Oracle Internet Directory configuration is valid.

Note: If OID Authenticator is the only provider, to ensure that the WebLogic domain starts properly, the WebLogic Server user account and its granted group memberships must be created in the Oracle Internet Directory.

13. On the **Provider Specific** tab, specify relevant values in the following fields:
 - **Host** – specify the host name of the Oracle Internet Directory.
 - **Port** – specify the port number associated with the Oracle Internet Directory.
 - **Principal** – specify an LDAP administrative user. For example, cn=orcladmin.
 - **Credential** – specify the password associated with the LDAP administrative user.
 - **Confirm Credential** – enter the password again to confirm the credential.
 - **User Base DN** – specify the distinguished name (DN) of the tree in the Oracle Internet Directory that contains the users.
 - **Use Retrieved User Name as Principal** – select this check box.
 - **Group Base DN** – specify the distinguished name (DN) of the tree in the Oracle Internet Directory that contains the groups.
 - **Propagate Cause For Login Exception** – select this check box.
14. Click **Save**.
15. The order in which providers populate a subject with principals is significant. You may want to reorder the list of all providers in your realm and bring the newly added provider to the top of the list, similar to the following:

- OSSO Identity Asserter
 - OID Authenticator
 - Default Authenticator
 - Default Identity Asserter
16. Save all configuration settings and restart the WebLogic server for the changes to take effect.
 17. Log on to the WebLogic Administration Console and navigate to the **Settings for myrealm** screen.
 18. Click the **Users and Groups** tab to view a list of users and groups included in the configured Authentication providers. You should see user names from the Oracle Internet Directory configuration, which verifies that the configuration is valid and working.
 19. If the Oracle Internet Directory instance is configured successfully, you can change the Control Flag:
 - If the Oracle Internet Directory authentication is sufficient for an application to identify the user, then choose the SUFFICIENT flag. SUFFICIENT means that if a user can be authenticated against Oracle Internet Directory, no further authentication is processed. The authentication will stop once the user successfully authenticates against the OID LDAP. This will make user IDs in the OID LDAP take precedence over any of the same names found in the embedded LDAP of the WebLogic Server.
 - If the OID LDAP contains all users, including the WebLogic administrative user account, then you may want to choose REQUIRED. REQUIRED means that the Authentication provider must succeed even if another provider already authenticated the user. This will force all user IDs to be validated against OID.

Note: In case the application requires the user names to be in the same case as stored in the Oracle Internet Directory, select the Use Retrieved User Name as Principal check box in the Provider Specific tab. See step 13.

20. Save and activate the changes.
21. Restart the WebLogic server.

Configure the Application for the OSSO Identity Asserter

The WebLogic Server supports adding multiple authentication-methods. If you are setting up an OSSO Identity Asserter in the WebLogic Application Console, the Web application using the OSSO Identity Asserter must have its auth-method set to CLIENT-CERT. After deploying the application on the WebLogic Server, all web.xml files in the application EAR file must include CLIENT-CERT in the element auth-method for the appropriate realm. To edit web.xml for the OSSO Identity Asserter:

1. Locate the **web.xml** file in the application EAR file. Since this file exists in a WAR file, you must first extract the WAR file, and then extract the web.xml file from the WAR. For example:


```
jar xvf myApp.ear myWebApp.war
jar xvf myWebApp.war WEB-INF/web.xml
```

2. In the web.xml file, locate the auth-method for the appropriate realm and enter CLIENT-CERT. For example:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>myRealm</realm-name>
</login-config>
```

If the application will be accessed outside of an OSSO environment, you can specify FORM or BASIC authentication method in the <auth-method> entry. For example:

```
<auth-method>CLIENT-CERT, FORM</auth-method>
```

3. Save the file.
4. Replace the web.xml file in the EAR archive.
5. Repeat the steps above for each web.xml file in the application EAR file.
6. Redeploy and restart the application.

Configuring Single Sign-On for Non-Web Retail Applications

This section describes the considerations for configuring single sign-on for the following Oracle Retail applications that are not Web-based systems:

Retail Merchandising System (RMS)

To set up a Oracle Forms-based application such as RMS for single sign-on, the Oracle Forms framework must be configured to use single sign-on. This can be achieved by enabling single sign on in the Forms framework configuration file (formsweb.cfg) and registering the mid-tier HTTP Server with Oracle Single Sign-On server. You must also take into consideration that the Forms framework uses Resource Access Descriptor (RAD) for mapping OSSO user IDs to the database connection strings.

For more information on setting up single sign-on, refer to the section *Setting Up RMS for Single Sign-On* in the *Oracle Retail Merchandising System Implementation Guide*.

Retail Price Management (RPM) and Store Inventory Management (SIM)

You can achieve single sign-on for RPM and SIM using the JnlpLaunch Servlet. The JnlpLaunch Servlet is a dynamically protected application. It authenticates the user with RPM or SIM using the single sign-on user name and temporary password. For more information, refer to the section *Review and/or Configure Oracle Single Sign-On* in the *Oracle Retail Price Management Installation Guide* or *Oracle Retail Store Inventory Management Installation Guide*.

References

This appendix provides a reference to relevant content in the Oracle Fusion Middleware documentation.

- Oracle White Paper – Extending WebCenter Spaces (11.1.1.2.0 and 11.1.1.3.0), June 2010, Section – Customizing Resource Catalogs. For more information on accessing this white paper, see [Accessing the Extending WebCenter Spaces White Paper](#).
- Oracle Fusion Middleware Developer's Guide for Oracle WebCenter 11g Release 1 (11.1.1), Chapter 6 – Configuring the Resource Catalog for Oracle Composer.
- Oracle Fusion Middleware Developer's Guide for WebCenter 11g Release 1 (11.1.1), Chapter 6 – Enabling Runtime Creation and Management of Pages.
- Oracle Fusion Developer's Guide for Oracle Application Development Framework 11g Release 1 (11.1.1), Part III – Creating ADF Task Flows.
- Oracle Fusion Middleware User's Guide for Oracle WebCenter 11g Release 1(11.1.1), Chapter 8 – Creating, Editing, and Deleting Pages.
- Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter 11g Release 1 (11.1.1), Chapter 22 – Managing External Applications (Section 22.2 – Registering External Applications).
- Oracle Fusion Middleware Developer's Guide for Oracle WebCenter 11g Release 1 (11.1.1), Chapter 37 – Securing Your WebCenter Application (Section 37.2 – Working with External Applications).
- Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter 11g Release 1 (11.1.1), Chapter 36 – Making Applications Available in WebCenter Spaces.



Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2008, Oracle. All rights reserved.
This document is provided for information purposes only and the contents hereof are subject to change without notice.
This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.