Oracle® Fusion Applications

Extensibility Guide

11*g* Release 1 (11.1.3)

E16691-04

March 2012

Documentation for business analysts and developers that describes how to customize and extend the standard functionality provided by Oracle Fusion Applications.

ORACLE®

Oracle Fusion Applications Extensibility Guide, 11*g* Release 1 (11.1.3)

E16691-04

Primary Author: Chris Kutler (lead), Sarah Bernau, Shelly Butcher, Ralph Gordon, Peter Jew, Mark Kennedy, Robin Whitmore, Steven Leslie, Landon Ott, Leslie Studdard, Marla Azriel

Contributing Author:

Contributor:

Contents

6 Customizing the Navigator Menu

7 Customizing and Extending BPMN Processes

8 Customizing Reports and Analytics

15 Customizing Security for ADF Application Artifacts

Preface

Welcome to *Oracle Fusion Applications Extensibility Guide.*

Audience

This document is intended for business analysts, administrators, and developers who want to customize and extend the standard functionality provided by Oracle Fusion Applications. Business analysts and administrators should have a basic understanding of Oracle Fusion Applications and Oracle Application Development Framework concepts and be familiar with the terms in the *Oracle Fusion Applications Master Glossary*. Developers should have a basic understanding of the Java programming language, web applications, Oracle JDeveloper, and Oracle Application Development Framework. This book gives an overview of the customization and extension tasks and provides references to the books that contain more detailed documentation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

You can also find information about Oracle Fusion Middleware and extending and customizing Oracle Fusion Applications in the following documents:

- *Oracle Fusion Applications Administrator and Implementor Roadmap*
- *Oracle Fusion Applications Common Implementation Guide*
- *Oracle Fusion Applications Developer's Guide*
- *Oracle Fusion Applications Concepts Guide*
- *Oracle Fusion Applications CRM Extensibility Guide*
- *Oracle Fusion Applications Master Glossary*

- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*
- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*
- *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*
- *Oracle Fusion Middleware User's Guide for Oracle Business Rules*
- *Oracle Fusion Applications Security Guide*
- *Oracle Fusion Applications Security Hardening Guide*
- *Oracle Fusion Middleware Application Security Guide*
- *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*
- *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*
- *Oracle Database Security Guide*
- *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*
- *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)*
- *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*
- *Oracle Fusion Middleware Developer's Guide for Oracle Enterprise Scheduling Service*
- *Oracle Fusion Applications Administrator's Guide*
- *Oracle Fusion Applications Patching Guide*
- *Oracle Fusion Applications Installation Guide*
- *Oracle Fusion Applications Administrator's Troubleshooting Guide*
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

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.

Convention	Meaning
italic	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.

What's New in This Guide for 11*g* Release 1 (11.1.3)

For 11*g* Release 1 (11.1.3), this guide has been updated in several ways. The following table lists the sections that have been added or changed.

For changes made to Oracle JDeveloper and Oracle Application Development Framework (Oracle ADF) for this release, see the What's New page on the Oracle Technology Network at http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html.

Sections	Changes Made
Chapter 1 Customizing and Extending Oracle Fusion Applications	
Section 1.2, "Understanding Customization Layers"	Revised section to remove mention of the obsolete Enterprise layer.
Chapter 2 Understanding the Customization Development Lifecycle	
Section 2.3.2, "Viewing Customizations Using the Manage Customizations Dialog"	Revised section to describe how to use the new feature to manage customizations for a user other than yourself.
Section 2.4, "Downloading and Uploading Customization Files"	Section added to describe how to download and upload customizations using either Oracle WebLogic Scripting Tool (WLST) commands, or Oracle Enterprise Manager Fusion Applications Control.
Chapter 3 Customizing Existing Pages	
Section 3.1, "About Customizing Existing Pages"	Section revised to introduce terms and provide better overview of Page Composer.
Section 3.1.3, "What You Can Do with Pages at Runtime"	Section revised to reflect feature changes. Source view is no longer available in CRM applications, except for editing the UI Shell template. Some tasks can only be completed in CRM dashboard pages or non-CRM applications.

Sections	Changes Made
Section 3.2, "Editing a Page in Page Composer"	Section revised to reflect feature changes. Source view is no longer available in CRM applications, except for editing the UI Shell template. Some tasks can only be completed in CRM dashboard pages or non-CRM applications. Task added to describe how to change the page title.
Chapter 5 Using Flexfields for Custom Attributes	
Section 5.3, "Planning Your Flexfields"	Section revised to describe new Highlight Flexfields and Page Composer features for determining whether there are flexfields on a page and determining a flexfield's name, code, module, and usage.
Section 5.5, "Configuring Flexfields"	Section revised to note naming restrictions for segment and context codes.
Chapter 7 Customizing and Extending BPMN Processes	
Section 7.5, "Deploying an Oracle BPM Project"	Revised section to include information on generating a deployment plan using Business Process Composer.
Section 7.6.2, "Configuring BPMN Processes in HCM Applications"	Added new section containing information on using BPMN processes in Oracle Fusion Human Capital Management applications.
Chapter 10 Using JDeveloper for Customizations	
Section 10.2.1, "Creating the Customization Application Workspace"	Section revised to reflect that the connection to the exploded EAR is now created in the Application Resources panel of the Application Navigator.
Section 10.2.5, "Resynchronizing Your Customization Workspace Configuration Files"	Section added to explain how to update local workspace configuration files after a patch to the application.
Chapter 11 Customizing and Extending ADF Application Artifacts	
Section 11.8, "Customizing and Extending the Oracle Fusion Applications Schemas"	Section added to describe how to make changes to Oracle Fusion Applications schemas.
Section 11.11, "Customizing Menus"	Section added to explain how to customize the home page, preferences, and navigator menus.
Chapter 14 Customizing and Extending Oracle Enterprise Scheduler Jobs	
Section 14.3, "Creating a Custom Oracle Enterprise Scheduler Application to Extend Oracle Enterprise Scheduler Jobs"	Section added to explain how to create, deploy, and register a customized hosting application for customizing and extending Oracle Enterprise Scheduler jobs.
Chapter 15 Customizing Security for ADF Application Artifacts	

Sections	Changes Made
Section 15.3.10, "What You Cannot Do with Security Policies at Design Time"	Revised section to remove recommendation to run GUID WLST command when migrating security policies to a production environment. Running the script is no longer necessary because reconciliation is now performed automatically during migration to a production environment.
Section 15.3.11, "Before You Begin Customizing Security"	Revised section to clarify that database resources should be created in a custom schema. Added a link to the chapter on Customizing and Extending ADF Application Artifacts for details.
Chapter 19 Customizing the Oracle Fusion Applications Skin	Chapter added to describe how to use Oracle Application Development Framework (ADF) Skin Editor to change the look and feel of Oracle Fusion applications.
Appendix A: Troubleshooting Customizations	Appendix added to discuss how to troubleshoot customization problems.

Part I

Introduction to Customizing and Extending Oracle Fusion Applications

Part I contains the following chapters:

1

Customizing and Extending Oracle Fusion Applications

This chapter provides an overview of how to customize and extend Oracle Fusion applications and, introduces the design time and runtime tools used in the process, such as Page Composer, CRM Application Composer, Oracle JDeveloper, Oracle SOA Composer, Business Process Composer, Oracle Business Intelligence (BI) Publisher, Oracle Business Process Management (BPM) Studio, Oracle BPM Worklist, Oracle Enterprise Manager Fusion Applications Control, and the Setup and Maintenance work area.

> **Note:** Before you begin to customize or extend Oracle Fusion applications, you first must install and implement them. For more information, see *Oracle Fusion Applications Administrator and Implementor Roadmap*.

This chapter includes the following sections:

1.1 Understanding Customizing and Extending Oracle Fusion Applications

While Oracle Fusion applications provide robust out-of-the-box functionality, there may be areas of one of the applications that you must change to meet your business needs. This book will guide you through the process of extending and customizing Oracle Fusion applications. Note that you can also create a complete Java EE application to supplement your Oracle Fusion applications. See the *Oracle Fusion Applications Developer's Guide* for more information.

All Oracle Fusion applications are based on Oracle Fusion Middleware. Most user interfaces are implemented using Oracle Application Development Framework (Oracle ADF) and standard Java technologies. The foundation of the applications are the Service Oriented Architecture (SOA) business processes. Business intelligence frameworks provide a number of reporting capabilities. Identity management works at every level to control access. Each of these areas of an application can be customized and extended to suit your business needs.

Additionally, Oracle Fusion applications are built using a common data model. Because of this commonality, when you make a customization in one area, that customization will be available to all objects in the application. For example, if you add an attribute to an object, you can easily add that attribute to the web-based view page, to an associated mobile page, and to any associated reports. And for the most part, the tools and processes you use to customize one application will be the same tools and processes to customize all of Oracle Fusion applications.

Note: If your Oracle Fusion applications are part of a multi-tenant environment, or is deployed as a Software as a Service Solution (SaaS) environment, then only a subset of customization capabilities will be available to you. See the SaaS documentation for your customization procedures.

Within this guide, the term *customizing* means to change a standard (existing) artifact. For example, you can add an attribute to a standard business object, or you can change what is displayed on a standard view page. The term *extending* means to create a completely new artifact, such as a custom business object or custom view page. For customizations and extensions, there are three basic scenarios: personalization, runtime customizations and extensions, and design time customizations and extensions.

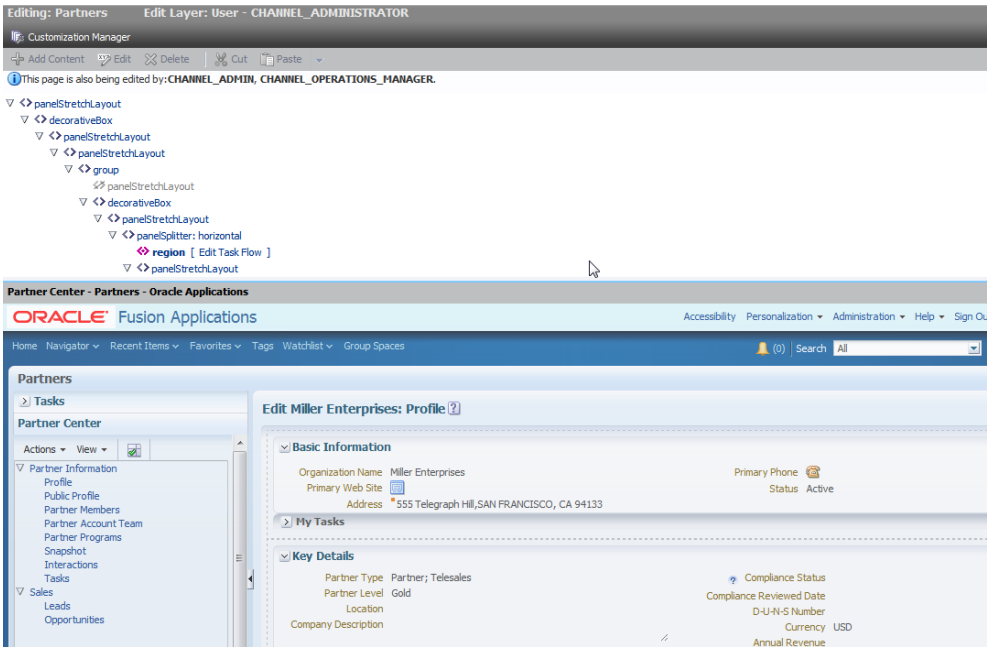
Personalization

Personalization refers to the changes that every end user of the Oracle Fusion Applications product suite can make to certain artifacts in the user interface (UI) at runtime. These changes remain for that user each time that user logs into the application. Personalization includes changes based on user behavior (such as changing the width of a column in a table), changes the user elects to save, such as search parameters, or composer-based personalizations, where an end user can redesign aspects of a page.

For composer-based personalizations, Oracle Fusion applications includes Page Composer, which allows end users to change certain UI pages to suit their needs. For example, they can rearrange certain objects on a page, add and remove designated content, and save queries. Figure 1–1 shows the Partner Profile page in Page Composer. An end user can add other content to this page, or change the order of the current content.

Note: By default, only certain personalizations are allowed. You can customize what can be personalized. For more information, see Chapter 17, "Configuring End User Personalization."

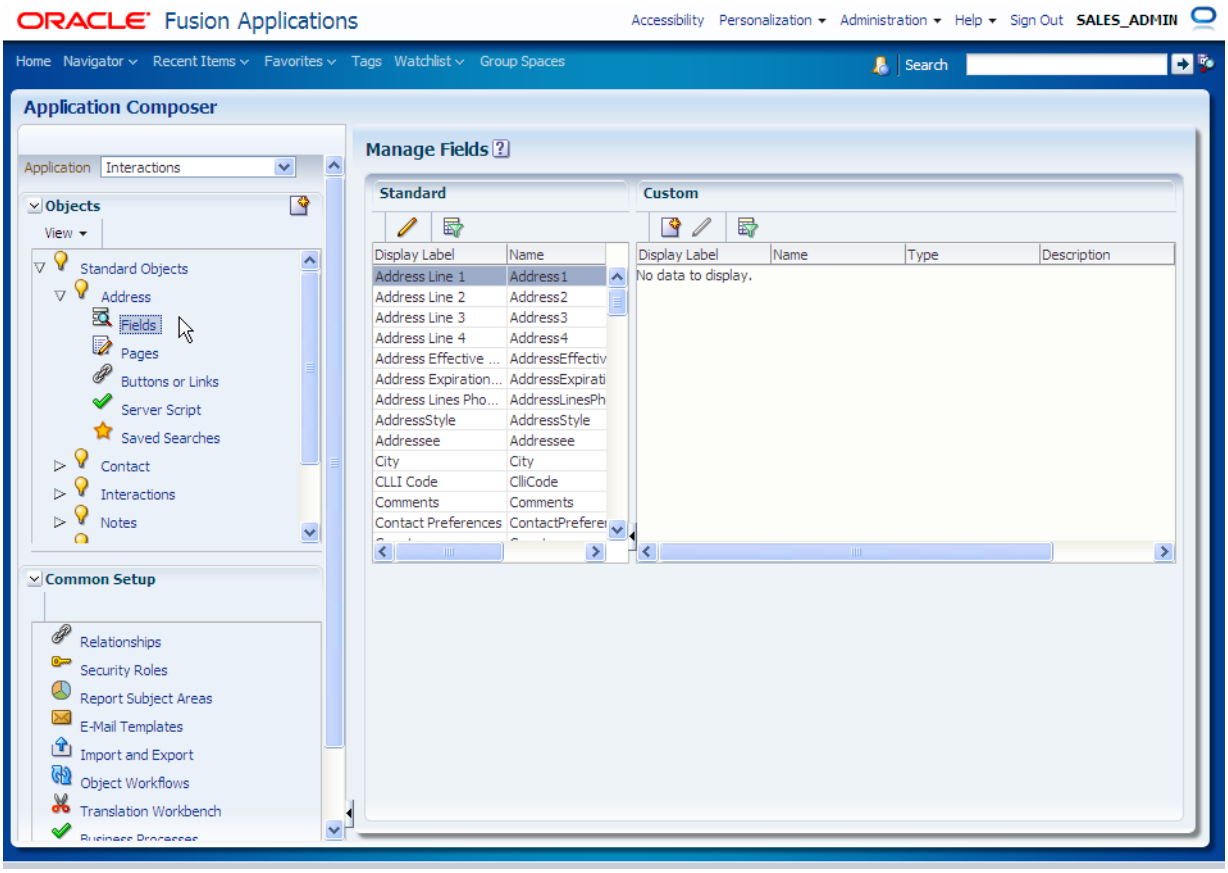
Figure 1–1 End Users Can Personalize UIs with Page Composer



Runtime customizations and extensions

Runtime customizations and extensions include those that a business analyst user at your enterprise can make to an Oracle Fusion application at runtime using browser-based composers. These customizations and extensions are surfaced to all or to a subset of Oracle Fusion Applications users, and range from changing the look and feel of a page, to customizing standard business objects, adding a new business object and associated pages and application functionality, changing workflows, defining security for new objects, and customizing reports. Figure 1–2 shows how you can customize the fields on a standard business object using Oracle CRM Application Composer, which is a runtime tool used to customize and extend certain CRM applications.

Figure 1–2 CRM Application Composer Allows You to Customize Business Objects at Runtime

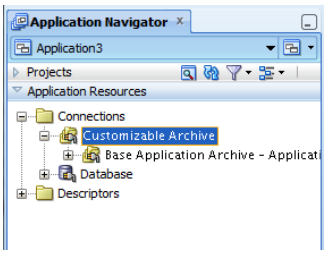


Customizing and extending Oracle Fusion applications using runtime tools are covered in Part II, "Business User Customizations and Extensions".

Design time customizations and extensions

Design time customizations and extensions include more complex changes, such as creating SOA composites or creating new batch jobs, and they require deployment into the runtime environment. Design time customizations are most often done by Java developers using Oracle JDeveloper (a comprehensive IDE), as shown in Figure 1–3, or they may be done in other tools, such as SOA Composer. The customizations are then uploaded or deployed to a running instance of Oracle Fusion Applications. Developer-level extensions are covered in Part III, "Developer Customizations and Extensions".

Figure 1–3 Oracle JDeveloper IDE



> **Tip:** You can also extend an Oracle Fusion application by creating a completely separate application and integrating it into Oracle Fusion Applications. For more information about creating an Oracle Fusion application, see the *Oracle Fusion Applications Developer's Guide*.

Most customizations made to an Oracle Fusion application, whether a personalization an end user makes, a change a business user makes using a runtime composer tool, or a change a developer makes using JDeveloper to create new source code, are stored in a business metadata repository. Because these customizations are kept separate from the base code, you can safely upgrade your Oracle Fusion application without overwriting or needing to redo your changes.

Customizations for the UI and for business components are created in layers, meaning that you can create them for specific users (as in the case of personalization), or for specific roles or sites, and the changes will be shown only when applicable. For more information about the metadata dictionary and customization layers, see Section 1.2, "Understanding Customization Layers."

Customizations made at runtime can be saved in a sandbox so that the changes can be isolated and validated before being published into a full test environment. Changes done at design time are done in a development environment, and can also be deployed to a sandbox before being deployed into the full test environment. For more information about sandboxes, see Section 2.2, "Using the Sandbox Manager."

Because most pages in an application consist of a number of different components (some of them actually being another page), it can be difficult to discern what has been customized, and for what layer. During customization of a page, you can use the Manage Customizations dialog to understand the state of each artifact on a page: whether it has been customized, and if so, for what layer. You can import customizations that others have done, or you can export your own customizations. For more information about using the Manage Customizations dialog and sandboxes, see Chapter 2, "Understanding the Customization Development Lifecycle."

1.2 Understanding Customization Layers

Oracle Fusion applications contain built-in customization layers that allow you to make customizations that affect only certain instances of an application. For example, the Sales application has a layer for job role. When you customize an artifact, you can choose to make that customization available only to users of a specific job role, for example, a sales representative.

Customizations you make are not saved to the base standard artifact. Instead, they are saved to an XML file that is stored in an Oracle Metadata Services (MDS) repository. This XML file acts like a list of instructions that determines how the artifact looks or behaves in the application, based on the layer that is controlling the current context. The MDS Customization Engine manages this process.

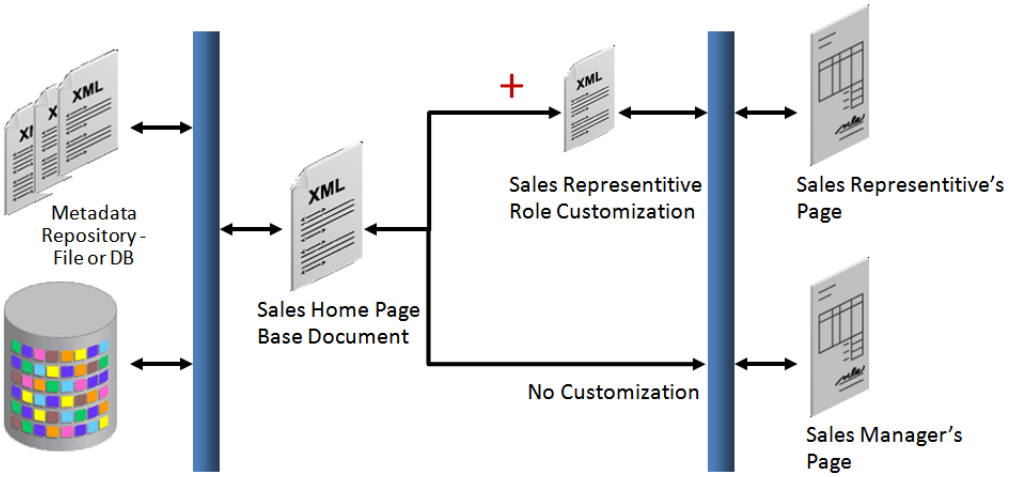
For example, say you want to customize the Sales home page by removing the Quick Create panel, but only for users with the Sales Representative role. Before you make your customization, you first select the layer to make your customization in, in this case the role layer whose value is `Sales Representative`. When you make your customization by removing that pane from the page, an XML file is generated with the instructions to remove the pane, but only in the role layer, and only when the value is `Sales Representative`. The original page file remains untouched. The MDS Customization Engine then stores the XML file in an MDS repository.

Now, whenever someone logs into the application and requests an artifact, the MDS Customization Engine checks the repository for XML files that match the requested

artifact and the given context, and if there is a match, it layers the instructions on top of the base artifact. In this example, whenever the Sales home page is requested (the artifact) by someone who is assigned the role of Sales Representative (the context), before the page is rendered, the MDS Customization Engine pulls the corresponding XML file from the repository and layers it on top of the standard Sales home page, thereby removing that pane. Whenever someone who is not a Sales Representative logs in (for example, someone with the role of Sales Manager), the XML file with your changes is not layered on top, and so the Quick Create panel is displayed.

Figure 1–4 shows how the customization XML file is applied to the base document and shown only to a sales representative.

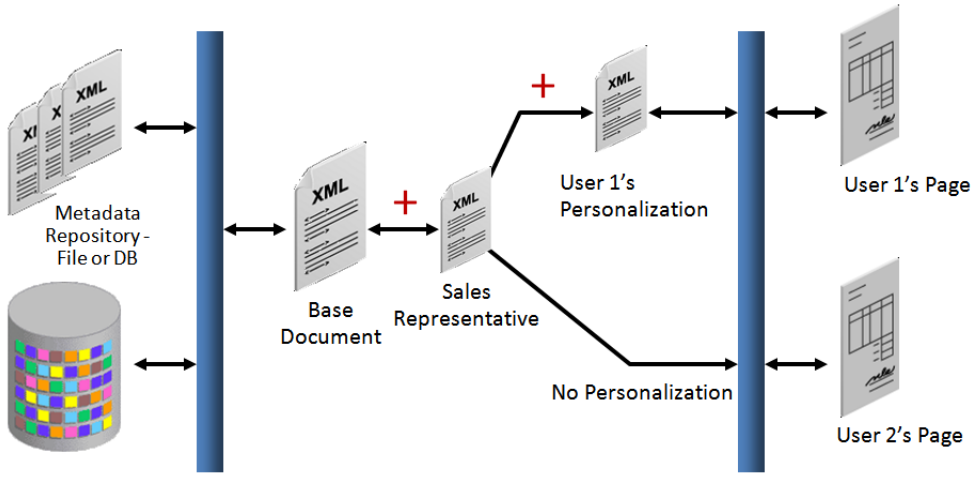
Figure 1–4 One Customization Layer Handled by the MDS Customization Engine



All users of Oracle Fusion applications can personalize certain pages using the Personalization menu. Users can move elements around on a page, hide elements, and even add available elements to their page. When they do this personalization, the MDS Customization Engine creates an XML file specific to that user.

For example, say User 1 (who has the role of Sales Representative) personalizes the Sales home page. There will then be an XML file stored in the repository, noting the changes that user made. When User 1 logs in, as in the previous example, the MDS Customization Engine pulls the XML file with the sales representative customizations from the repository and layers it on top of the standard Sales home page. In addition, the engine pulls the XML file with User 1's personalizations, allowing the user to see the personalization changes along with the Sales Representative changes. When other Sales Representatives log in, they do not see User 1's personalization changes, as shown in Figure 1–5.

Figure 1–5 Personalizations Are Also Handled by the MDS Customization Engine



The exact customization layers available for an application depend on that application family (see the product-specific documentation from Oracle Enterprise Repository for Oracle Fusion Applications for details). However, all Oracle Fusion applications have the following customization layers:

- Global: When customizations are made in this layer, they affect all users of the application. This layer's XML files are added for everyone, whenever the artifact is requested. Customizations made to ADF Business Components in JDeveloper must be made in the Global layer.
- Site: Customizations made in the Site layer affect users at a particular location.
- User: This is where all personalizations are made. Users do not have to explicitly select this layer. It is automatically selected when you use the Personalization menu.

These layers are applied in a hierarchy, and the highest layer in that hierarchy in the current context is considered the *tip*. Within the default customization layers, the Global layer is the base layer, and the User layer is the tip. If customizations are done to the same object, but in different layers, at runtime, the tip layer customizations take precedence. For example, if you customize the label for a field in the site layer using Page Composer and customize the same label in the global layer using JDeveloper, the site layer customization will be displayed at runtime.

Because customizations are saved in these XML files, when you patch or upgrade your Oracle Fusion applications, the base artifacts can be updated without touching your changes. The base artifact is replaced, and when the application is run after the patch or upgrade, the XML files are simply layered on top of the new version. You do not need to redo your customizations.

Before you create customizations, you must select the layer to which you want your customizations to be applied. Most of the tools you use to create your customizations provide a dialog where you can pick the layer for your customizations.

1.3 Understanding the Business User and Developer Tools

The user interfaces in Oracle Fusion applications are controlled by role based authentication, meaning that the information presented in the UI, and what the user can do in the UI, depends on the role assigned to the currently logged in user. For example, if you are assigned a role with administrative privileges, when you log in to

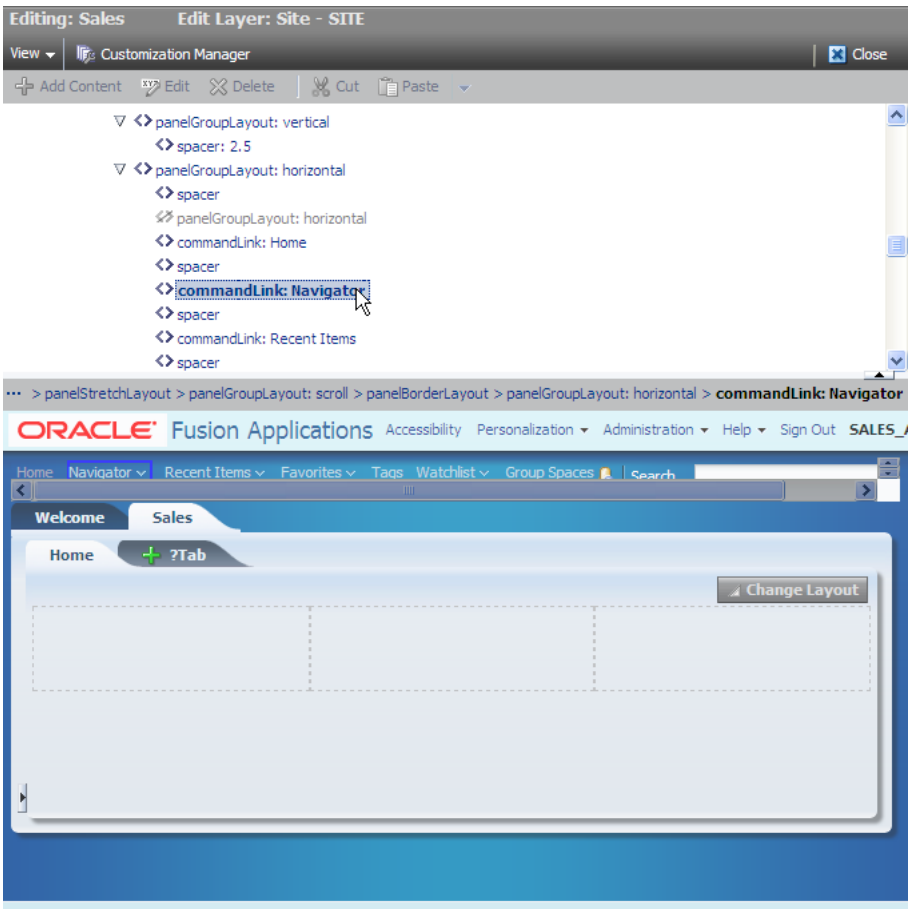
Oracle Fusion Applications, you will see an Administration menu, as shown in Figure 1–6. This menu allows you to do things like customize a page for all users, or manage customizations.

Figure 1–6 Oracle Fusion Applications Menu Bar

Both personalization and customization use Page Composer to make changes to an application page. Using personalization, any user can drag and drop fields, rearrange regions, add approved external content and save their favorite queries.

Using administration customization, you also use Page Composer to customize pages for other users. You can add fields, add validation, change defaults, rearrange regions, and add external content. Page Composer allows you to work in either a WYSIWYG view or source view, as shown in Figure 1–7.

Figure 1–7 Page Composer



For more information about customizing pages, see Chapter 3, "Customizing Existing Pages."

If you need to extend or customize the Sales, Marketing, Customer Center, Trading Community Architecture (TCA), and Order Capture applications that are part of the Customer Relationship Management (CRM) product family of Oracle Fusion Applications, then you can use CRM Application Composer to customize your pages.

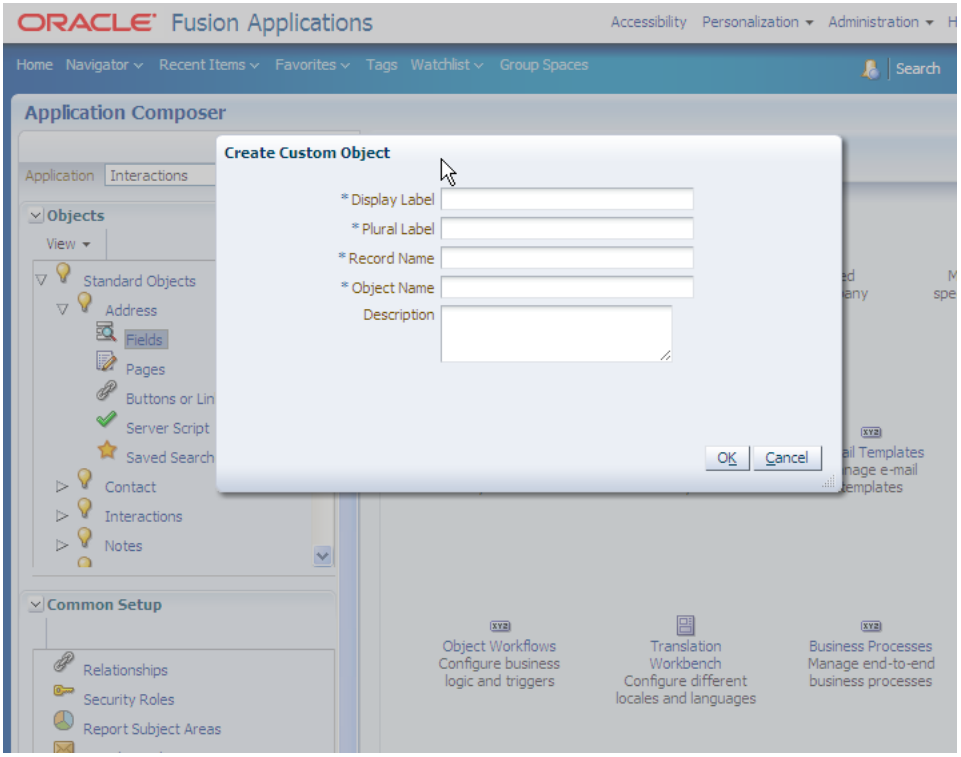
Note: Only certain pages are available for customization. For a complete list, see the product-specific documentation from Oracle Enterprise Repository for Oracle Fusion Applications.

You access CRM Application Composer by clicking the **Application Composer** link from the Navigator menu of Oracle Fusion Applications, as shown in Figure 1–8.

Figure 1–8 Navigator Menu

CRM Application Composer also allows business analysts to make more complex runtime customizations. In addition to customizing pages, business analysts can customize business objects and all the artifacts that support them (such as fields, pages, buttons and links, security, server scripts, and saved searches), and can also extend Oracle Fusion applications by creating completely new business objects and artifacts, as shown in Figure 1–9. For more information, see Chapter 4, "Customizing Objects."

Figure 1–9 CRM Application Composer



When new business objects are created, you often also create associated Work Area pages for those objects. You can add those pages to the Navigator menu so that they can be accessed in the same way as standard objects. For more information, see Chapter 6, "Customizing the Navigator Menu."

When you create a new top-level business object, you can also create a new object workflow to manage any business processes associated with it. For example, say you used CRM Application Composer to create a marketing object and you want to create an associated approval flow. From within CRM Application Composer, you can access Oracle Business Process Composer and create the process that defines that flow. For applications outside of CRM, you access Business Process Composer directly from the Navigator menu. For more information about using the Business Process Composer, see Chapter 7, "Customizing and Extending BPMN Processes."

When you create a new object in CRM Application Composer, you can define security policies for it. Data security policies define the end user's level of access to the data records of the business object. Function security policies define the end user's level of access to the application resources that display the object (for example a page or a task flow). Both types of security must be defined for a security policy to be complete. Security policies are not stored in an MDS repository. Instead, they are stored in Oracle Fusion Data Security database tables (for data security) and in an LDAP-based server running Oracle Internet Directory (for function security). For more information about creating security policies for custom CRM business objects, see Chapter 9, "Customizing Security for Custom Business Objects."

If you need to add an attribute to an object in an application that is not one of the five CRM applications, you can often use *flexfields*. Flexfields allow you to define object attributes on objects and then apply business logic to them. For example, an airline manufacturer might require very specific attributes for their orders that are not provided by the out-of-the-box implementation of an order. Because a flexfield exists for the order object, you can use it to create and configure the desired attribute. Flexfield configurations are stored in an MDS repository, and so are safe during patching and upgrading. You access flexfields from the Setup and Maintenance menu item from the Administration menu. For more information about flexfields, see Chapter 5, "Using Flexfields for Custom Attributes."

Oracle Fusion Applications come with a complete set of reports. You can customize these reports (for example, change the layout) to fit your particular business needs. Additionally, if you customize or create a business object, you can create a new report for that object. For more information, see Chapter 8, "Customizing Reports and Analytics."

To customize or create business objects outside of the five CRM applications, or when required customizations cannot be made in one of the runtime composers, use JDeveloper. When you work in a JDeveloper environment, you create a workspace that contains your changes and additions. When you create this workspace, you do so in the Oracle Fusion Applications Developer role. Like Oracle Fusion applications, JDeveloper uses roles to shape what you see and can do in the IDE. Work done in a developer role is stored in actual projects with code that gets deployed to an environment. Use the Oracle Fusion Applications Administrator Customization role when customizing an existing standard object (as opposed to creating a new object). Work done in this role is saved to an XML file that gets deployed into an MDS repository, keeping your changes separate from the base code. For more information about how to set up your JDeveloper customization environment, see Chapter 10, "Using JDeveloper for Customizations."

Note: You cannot create your own roles to define what you see and what you can do in JDeveloper.

Developers can use JDeveloper to create and customize view pages, business objects, task flows (flows that control navigation and business logic within the application), searches, and resource bundles. All customizations and extensions created in JDeveloper must be deployed to an environment. For more information about using JDeveloper to customize business objects and associated artifacts, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

SOA composites are the foundation on which Oracle Fusion applications are built: they are the glue that holds all the different application artifacts together and they allow the different applications to work in a unified manner. SOA composites for an application contain artifacts like Business Process Execution Language (BPEL) process flows. These flows provide communication between applications, additional human-based workflows, and business rules that determine branching in those flows. Developers can customize existing composites or create new ones using a mixture of JDeveloper and browser-based tools. Customized and extended SOA composites are all stored in MDS repositories. For more information, see Chapter 12, "Customizing and Extending SOA Components."

Some Oracle Fusion applications provide business process modeling (BPM) project templates that you can use to create BPM projects. BPM projects consist of SOA artifacts, such as business rules and human tasks, and Business Process Modeling Notation (BPMN) processes. You can customize these project templates to suit your business needs. For more information, see Chapter 13, "Customizing and Extending Oracle BPM Project Templates."

Finally, when you create custom pages, you may want to make them personalizable, so that end users can change the page for themselves. For more information, see Chapter 17, "Configuring End User Personalization." Also, when you make any type of customization or extension to Oracle Fusion applications, you might have to change the embedded help that appears on the screen. For more information, see Chapter 18, "Customizing Help." And all customizations can be translated. For more information, see Chapter 16, "Translating Custom Text."

Tip: When you extend Oracle Fusion applications, you may want those extensions to be configurable using Oracle Fusion Functional Setup Manager. For more information about creating setup flows for extensions, see the Oracle Fusion Applications Information Technology Management, Implement Applications Guide.

For a more detailed description of the workflow you must follow when customizing and extending, see Chapter 2, "Understanding the Customization Development Lifecycle."

1.3.1 What You Can Customize and Extend and with Which Tool

Table 1–1 shows the artifacts that you can customize or create in Oracle Fusion Applications, what tool you use, the type of user that can make the change, and whether the changes are stored in an MDS repository.

Note: CRM Application Composer is only available if you want to make changes in the following CRM applications:

- Marketing
- Sales
- Customer Center
- Trading Community Architecture (TCA)
- Order Capture

Note: While you can customize view pages in Page Composer and CRM Application Composer, only certain pages are configured to allow it. If the customization you want to make is not available in Page Composer, then you must use JDeveloper to make the customization.

Table 1–1 Customization Scenarios in Oracle Fusion Applications

Customization/ Extension	Tool	Type of User	MDS?	Where to Find Information
View Pages				
Add, move, and delete components on a page.	Page Composer	Business Analyst	Yes	Section 3.2, "Editing a Page in Page Composer"
Add fields, buttons, links, to a standard page (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.3, "Editing a Page in CRM Application Composer"
Customize properties on UI components on a standard page	Page Composer	Business Analyst	Yes	Section 3.3, "Editing Component Properties in Page Composer"
Customize properties on UI components on a standard page (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.2, "Editing Objects"
Make UI components on a page personalizable	Page Composer	Business Analyst	Yes	Section 17.3, "Configuring End User Personalization for Components"
Customize the UI Shell template	JDeveloper	Developer	Yes	Section 11.10, "Editing the UI Shell Template"
Customize the UI Shell template (CRM)	Page Composer	Business Analyst	Yes	Section 3.4, "Editing the UI Shell Template Used by All Pages"
Define resource bundles	JDeveloper	Developer	Yes	Section 11.12, "Customizing or Adding Resource Bundles"
Make a custom page personalizable (custom pages created in CRM Application Composer are customizable by default)	JDeveloper	JDeveloper	Yes	Section 17.2, "Allowing Pages to be Personalized by End Users in Page Composer"

Table 1–1 (Cont.) Customization Scenarios in Oracle Fusion Applications

Customization/ Extension	Tool	Type of User	MDS?	Where to Find Information
Customize onscreen hoover help text	Page Composer	Business Analyst	Yes	Section 18.3, "Customizing or Adding Bubble Embedded Help"
Customize onscreen help text	JDeveloper	Developer	Yes	Section 18.4, "Customizing or Adding Static Instructions, In-field Notes, and Terminology Definitions"
Change the look and feel of the entire application	JDeveloper	Developer	No	You can choose the skin you want Oracle Fusion Applications to use. For more information, see Chapter 19, "Customizing the Oracle Fusion Applications Skin."
Translate custom text	JDeveloper	Developer	Yes	Chapter 16, "Translating Custom Text"
Business Objects				
Customize business objects	JDeveloper	Developer	Yes	Section 11.2, "Editing Existing Business Components"
Customize business objects (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.2, "Editing Objects"
Add an attribute to a business object using flexfields (not CRM)	Setup and Maintenance work area	Business Analyst	No	Chapter 5, "Using Flexfields for Custom Attributes"
Create business objects	JDeveloper	Developer	Yes	Section 11.5, "Creating Custom Business Components"
Create business object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.4, "Creating Custom Objects"
Add business object page to Navigator menu	Setup and Maintenance work area	Business Analyst	No	Chapter 6, "Customizing the Navigator Menu"
Add custom business object work area pages to Navigator menu (CRM)	CRM Application Composer	Business Analyst	No	Section 4.4, "Creating Custom Objects"
Add validation to an object	JDeveloper	Developer	Yes	Section 11.5, "Creating Custom Business Components"
Add validation to an object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.2, "Editing Objects"
Customize saved search for a custom object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.5, "Creating and Editing Search Objects"
Create search for an object	JDeveloper	Developer	Yes	Section 11.9, "Customizing or Creating a Custom Search Object"
Create saved search for a custom object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.5, "Creating and Editing Search Objects"
Customize task flows for an object	JDeveloper	Developer	Yes	Section 11.3, "Editing Task Flows"

Table 1–1 (Cont.) Customization Scenarios in Oracle Fusion Applications

Customization/ Extension	Tool	Type of User	MDS?	Where to Find Information
Create task flows for an object	JDeveloper	Developer	Yes	Section 11.6, "Creating Custom Task Flows"
Customize object workflows for an object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.2, "Editing Objects"
Create object workflows for an object (CRM)	CRM Application Composer	Business Analyst	Yes	Section 4.4, "Creating Custom Objects"
Business Processes				
Create BPMN process in a BPM project	Business Process Composer	Business Analyst	Yes	Section 7.2, "Creating an Oracle BPM Project"
Create BPMN approval process in a BPM project (CRM)	CRM Application Composer	Business Analyst	Yes	Section 7.2, "Creating an Oracle BPM Project"
Customize custom BPM projects	Business Process Composer	Business Analyst	Yes	Section 7.3, "Customizing BPMN Processes"
Customize custom BPM projects (CRM)	CRM Application Composer	Business Analyst	Yes	Section 7.3, "Customizing BPMN Processes"
Customize BPM project templates	Oracle BPM Studio	Developer	Yes	Chapter 13, "Customizing and Extending Oracle BPM Project Templates"
Customize BPEL process or Mediator component, or add additional SOA components	JDeveloper	Developer	Yes	Section 12.4, "Extending or Customizing Custom SOA Composites"
Customize task routing rules, business rules, DVM and, composite properties	Process Workspace, Oracle SOA Composer and Fusion Applications Control	Developer	Yes	Section 12.2, "Customizing SOA Composites"
Reports				
Create report layout	Oracle BI Publisher	Business Analyst	No	Section 8.2.2, "Customizing Layouts"
Customize report layouts	Oracle BI Publisher	Business Analyst	No	Section 8.2.2, "Customizing Layouts"
Customize style templates	Oracle BI Publisher	Business Analyst	No	Section 8.2.2, "Customizing Layouts"
Create a report	Oracle BI Publisher	Business Analyst	No	Section 8.2.4, "Creating Custom Reports"
Translate a report	Oracle BI Publisher	Business Analyst	No	Section 8.2.5, "Adding Translations"
Create report subject area (CRM)	CRM Application Composer	Business Analyst	No	Section 4.4, "Creating Custom Objects"
Analyses and Dashboards				
Customize analytics	Reports and Analytics pane	Business Analyst	No	Section 8.3.2, "Customizing Analytics"

Table 1–1 (Cont.) Customization Scenarios in Oracle Fusion Applications

Customization/ Extension	Tool	Type of User	MDS?	Where to Find Information
Customize and extend the Oracle BI repository (RPD file)	JDeveloper	Developer	No	Section 8.3.3, "Customizing the Oracle BI Repository (RPD)"
Enterprise Scheduler Jobs				
Create Jobs	JDeveloper	Developer	No	Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs"
Security				
Add data security to custom object	Setup and Maintenance work area	Developer	No	Section 15.4, "Defining Data Security Policies on Custom Business Objects" and Section 15.5, "Enforcing Data Security in the Data Model Project"
Add function security to custom object	JDeveloper	Developer	No	Section 15.6, "Defining Function Security Policies for the User Interface Project"
Add security to custom object (CRM)	CRM Application Composer	Business Analyst	No	Section 9.2, "Defining Security Policies for Custom Business Objects"
Enable end users to set instance level security	CRM Application Composer	Business Analyst	No	Section 9.3, "Enabling End User Instance-Level Security Customization"

1.3.2 Installing Customization Tools

All the business analyst tools are available from the Navigator menu of Oracle Fusion Applications. However, developers must install and configure JDeveloper. After installing JDeveloper, they must set up their environment for customization and extending.

For procedures for installing JDeveloper and setting it up for extending (that is, for creating new objects), see the "Setting Up Your Development Environment" and "Setting Up Your JDeveloper Workspace and Projects" chapters in the *Oracle Fusion Applications Developer's Guide*.

For procedures for setting up JDeveloper for customizations, see Chapter 10, "Using JDeveloper for Customizations."

2

Understanding the Customization Development Lifecycle

This chapter discusses the typical workflow for customizing and extending Oracle Fusion applications. It describes how to use sandboxes to perform customizations in a segregated environment, publish the changes to a full test environment, and export the changes to other environments.

This chapter includes the following sections:

2.1 Understanding Typical Customization Workflows

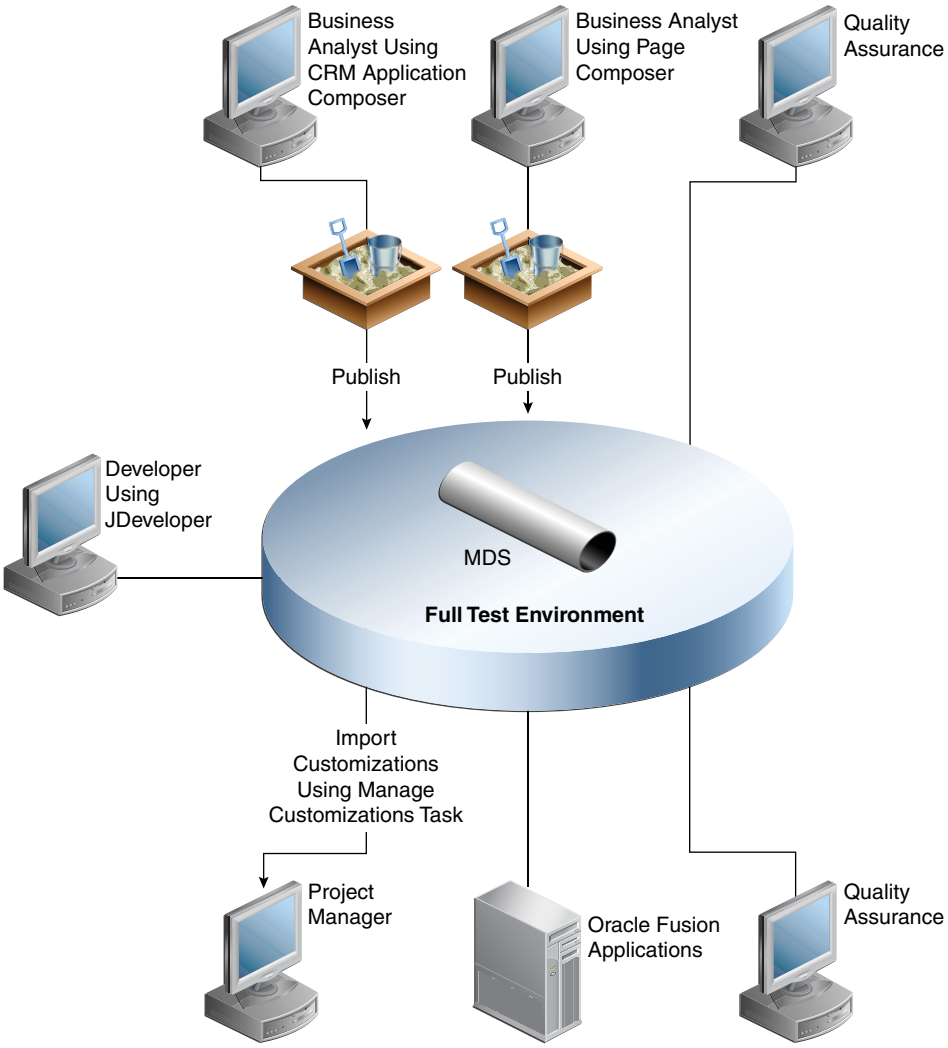
All customizations and extensions to Oracle Fusion Applications, whether done by business analysts or developers, should be done in a test environment. Typically, this environment contains one or more Oracle Fusion applications that will then be moved to a production environment after all customizations and extensions are complete and tested.

As described in Section 2.1.1, "Runtime Customization Workflow," business analysts using Page Composer and CRM Application Composer can make application customizations in *sandboxes*. Sandboxes store the customizations in Extensible Markup Language (XML) files in a separate Oracle Metadata Services (MDS) repository that is available only when you work in that particular sandbox. The changes can be done in a test-only sandbox (that is, the code in the sandbox is for testing only, and is never deployed), or they can be done in a sandbox that is then published to the environment.

Developers using design time tools, such as Oracle JDeveloper, can deploy their customizations directly to that environment, or they can publish to a sandbox, as described in Section 2.1.2, "Design Time Customization Workflow."

Project managers can monitor the customizations, and can also import and export customizations. The entire environment with all customizations can then be tested, as shown in Figure 2–1.

Figure 2–1 Customization Workflow in a Full Test Environment



Tip: When you extend Oracle Fusion applications, you might want users to be able to configure the extensions using Oracle Fusion Functional Setup Manager. For more information about creating setup flows for extensions, see the *Oracle Fusion Applications Information Technology Management, Implement Applications Developer Guide*.

2.1.1 Runtime Customization Workflow

When you use CRM Application Composer and Page Composer to make runtime customizations to Oracle Fusion applications, you can use sandboxes to save your changes in a segregated environment. For example, before you begin making customizations, you create a sandbox named `MySandbox` and make your customizations in that sandbox. If others want to see the customizations, then they would use `MySandbox`.

Note: There are restrictions for when more than one user is working in a sandbox. For more information, see Section 2.2.1, "Sandboxes and Concurrent Usage."

You can also use a sandbox when you define security policies for custom objects that you have created using CRM Application Composer. A security sandbox stores the security information in new database tables that are available only when you choose to work in that sandbox.

After you complete your customizations, the sandbox can be reviewed and approved by others, and then *published* to the full test environment where your customizations become part of that repository. For more information about sandboxes, see Section 2.2, "Using the Sandbox Manager."

Note: Flexfield sandboxes are for testing only and cannot be published. Instead, you *deploy* a flexfield to the full test environment using the flexfield UI. To test a flexfield configuration before deploying it to the full test environment, deploy it to a flexfield sandbox, as described in Section 5.7, "Deploying Flexfield Configurations." The changes that you deploy to a sandbox are isolated from the full test environment and can be seen only by those who make the flexfield sandbox active in their session. After you are satisfied with the changes in the sandbox, you can deploy the changes to the full test environment.

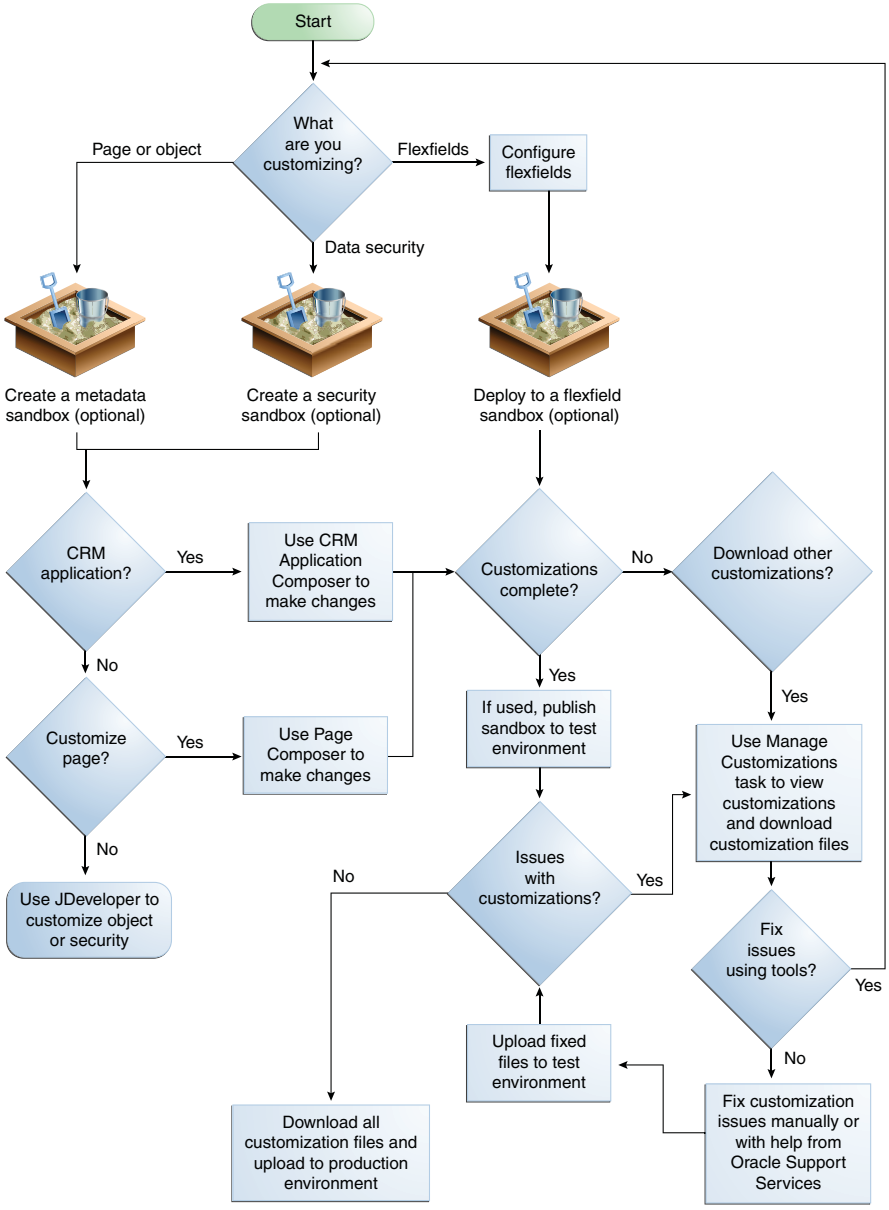
When you publish a sandbox, the published customizations are labeled. Labeling can act as a *save point*, meaning that if a future customization causes issues, you can use the Manage Customizations dialog to remove all customizations done after that point by *promoting* the last known good label back to the *tip*. For more information, see Section 2.3.3, "Backing Out Customizations."

You can also use the Manage Customizations dialog to view others' customization metatdata files, and to download those files to manually move them to another environment or to diagnose any issues. You can also upload others' customization metadata files to your environment. For more information, see Section 2.3, "Viewing and Diagnosing Runtime Customizations."

Note: Navigator menu, BPMN process flows, and report customizations do not use an MDS repository. Refer to Chapter 6, "Customizing the Navigator Menu," Chapter 7, "Customizing and Extending BPMN Processes," and Chapter 8, "Customizing Reports and Analytics" for more information.

Figure 2–2 illustrates the use of sandboxes when customizing pages, objects, and security using Page Composer and CRM Application Composer and when configuring flexfields.

Figure 2–2 Typical Runtime Customization Workflow



2.1.2 Design Time Customization Workflow

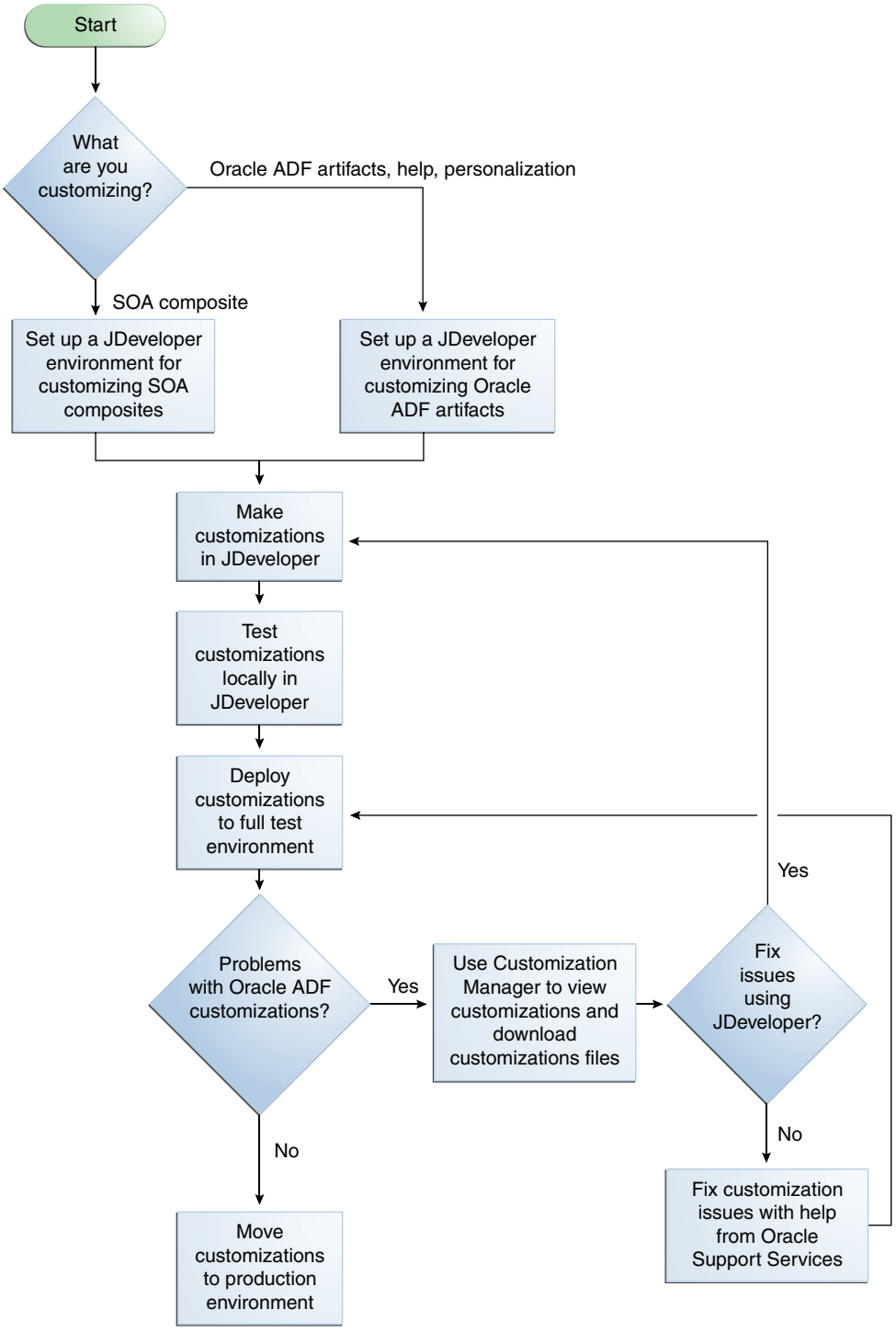
As shown in Table 1–1, some types of customizations, such as user interface template customizations, must be made at design time using JDeveloper. After you create these customizations, you can test them locally in JDeveloper and then deploy them directly into the full test environment. You can also deploy your customizations to a sandbox. Note that security customizations done at design time are not saved to a sandbox. Additionally, you can use source control software to manage design-time customizations. For more information about what source control software JDeveloper supports, see the "Versioning Applications with Source Control" topic of the JDeveloper online help.

Because your customizations (other than security changes) are stored in customization XML files in an MDS repository, they can also be viewed and managed using the Manage Customizations dialog.

Note: Unlike runtime customizations, you cannot promote design time customizations, as described in Section 2.3.3, "Backing Out Customizations." If a design time customization is causing a problem, you must undeploy the changes, fix the issues, and then redeploy to the test environment.

Figure 2–3 shows the flow for a typical design time customization process.

Figure 2–3 Typical Design Time Customization Workflow



2.2 Using the Sandbox Manager

The sandbox manager is a tool for managing the different types of customization changes that can be applied to an application. The different types of sandboxes are:

- Metadata

  The metadata sandbox supports making changes to the application's metadata stored in the MDS repository.

- Security

  The security-enabled sandbox supports making data security changes.

- Flexfield

  The flexfield sandbox is not created using the sandbox manager. Use the flexfield UI to make changes to the flexfields and then deploy them to the sandbox. The flexfield deployment process manages the creation of the sandbox. For more information about flexfields, see Section 5.7, "Deploying Flexfield Configurations."

To customize an Oracle Fusion application in runtime, you first create a sandbox and then use Page Composer or CRM Application Composer to make the customizations. These changes will be contained within the sandbox so they do not affect the mainline code. You then test and validate the changes by publishing the sandbox to the full test environment. After the application has been tested, it can then be moved to the production environment where the customization changes will be available to users of the system.

You can make changes to an application at runtime in a sandbox so that the changes are isolated from the mainline code. The mainline code is a branch of data that serves as a single source of truth. After you are satisfied with the changes in the sandbox and want to commit them, you can publish the metadata or security-enabled sandbox to the mainline code. Flexfield sandboxes are for testing only and cannot be published. You make flexfield configurations that are then stored in a database, and then deploy those configurations to a sandbox to see the resulting deployment artifacts in a sandbox environment. Flexfields are deployed directly to the mainline code using the flexfield UI. For more information about flexfields, see Section 5.7, "Deploying Flexfield Configurations."

You can use runtime tools to customize the application. The sandbox manager works with CRM Application Composer and Page Composer to customize objects and pages:

- For information about using CRM Application Composer, see Chapter 4, "Customizing Objects."
- For information about using Page Composer, see Chapter 3, "Customizing Existing Pages."

Oracle Business Process Composer and Oracle SOA Composer are also runtime customization tools, but they do not use the sandbox manager. They have their own mechanisms for handling customization changes:

- For information about using Oracle Business Process Composer, see Chapter 7, "Customizing and Extending BPMN Processes."
- For information about using Oracle SOA Composer, see Chapter 12, "Customizing and Extending SOA Components."

The metadata sandboxes created using the sandbox manager are available in JDeveloper when you are creating and deploying customizations intended for a deployed Oracle Fusion application in Oracle WebLogic Server. The available sandboxes will appear in a selection list in JDeveloper during deployment. For more information, see Section 11.13, "Deploying ADF Customizations and Extensions." Note that the security sandboxes created using the Sandbox Manager are not available in JDeveloper.

In CRM Application Composer, you can customize security policies using a sandbox specifically for editing them. When you enable the security sandbox, the operation will duplicate the schema for Oracle Fusion Data Security tables, which is a lengthy setup operation. Therefore, ensure that you are fully ready to customize the security policies *before* you enable the security sandbox.

The security policy customizations that you publish from a sandbox will be merged into the Oracle Fusion security policy repository as part of the native application, and will overwrite any previous customizations. Ensure that you are not editing the same object concurrently with another user to avoid inconsistencies can result when multiple users edit the security policies associated with the same object in different sandboxes.

The metadata and security sandbox sessions can be saved, downloaded, and imported as files into other Oracle Fusion applications.

2.2.1 Sandboxes and Concurrent Usage

In the customization runtime workflow, sandboxes are used to isolate the changes from the mainline code for testing and validating. After you are satisfied with the changes, you can publish the changes back to the mainline code. You can also create sandboxes for testing purposes only, and not publish them.

There are two types of sandboxes:

- Sandboxes intended to be published.

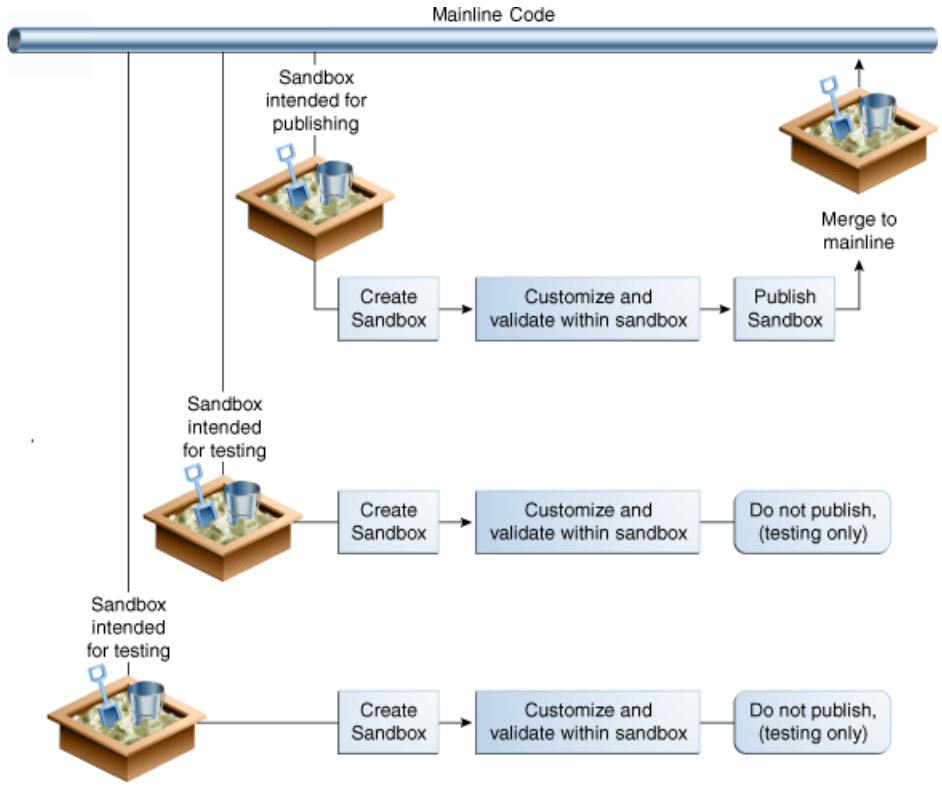
  These sandboxes will be merged back to the mainline code.

- Sandboxes intended for *test-only* purposes.

  These test-only sandboxes will not be published and therefore produce no concurrency conflicts between sandboxes. You can have many test-only sandboxes at the same time. But if you have multiple users working on the same test-only sandbox, then they must adhere to the guidelines described in Section 2.2.1.3, "Guidelines for One Sandbox, Multiple Users."

Figure 2–4 illustrates the two types of sandboxes and their relationship to the mainline code.

Figure 2–4 Two Types of Sandboxes



When multiple users can customize an application using sandboxes, there are two types of concurrency conflicts:

- Conflicts within a sandbox: Users overwriting changes created by other users, either directly by changing the same artifact, or indirectly by affecting files that are shared between the artifacts. For more information, see Section 2.2.1.3, "Guidelines for One Sandbox, Multiple Users."
- Conflicts between sandboxes (intended for publishing only): Multiple sandboxes with the same customized artifact publishing to the mainline code. For more information, see Section 2.2.1.4, "Guidelines for Multiple Sandboxes, Multiple Users."

An application artifact typically includes several metadata files. Therefore, creating or editing an artifact usually means making changes, whether directly or indirectly, to more than one file. Some of these metadata files may be shared between artifacts.

Note: Many customization scenarios, including customizing security policies, involve editing the same underlying artifacts. The only way to be certain of eliminating the possibility of any concurrency conflicts is to allow only one user at a time in an active sandbox. If you must have multiple users or multiple sandboxes, then follow the usage guidelines described in Section 2.2.1.3, "Guidelines for One Sandbox, Multiple Users," and Section 2.2.1.4, "Guidelines for Multiple Sandboxes, Multiple Users."

2.2.1.1 Conflicts Within a Sandbox

Conflicts within a sandbox can arise when multiple users are customizing an application using the same sandboxes at the same time, because more than one user may be attempting to customize the same artifact, or performing a customization task that indirectly affects other shared files. An example of a *direct conflict* is when different users attempt to customize the same page, the same fragment, or the same metadata file within the same layer. An example of an *indirect conflict* is when two users, each creating their own object, cause a conflict in the metadata file that tracks which new objects have been created by both saving their changes around the same time.

Conflicts may also arise when users are editing a shared artifact, such as when a user performs an operation that adds or edits a translatable string. For example, a user edits a field's display label or help text, or a validation rule's error message, while another user performs an operation around the same time that similarly affects translatable strings. Another example of a shared artifact conflict is when two or more users are working in navigator menus that are shared across applications.

When you are using Page Composer, although you are not required to, you should use a sandbox. Page Composer provides concurrency warning messages when there is a potential customization conflict with another user. For Page Composer, these warning messages help prevent conflicts. For more information, see the "Editing Pages" section of the *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

When you are using CRM Application Composer, always use a sandbox. CRM Application Composer provides error messages when there is a concurrency conflict. If multiple users in a single sandbox attempt to edit the same artifact, then an error message is displayed. For example, if two users are editing the label of the same attribute, then CRM Application Composer will display an error message.

2.2.1.2 Conflicts Between Sandboxes

Conflicts between sandboxes can arise when there is more than one sandbox intended for publishing in use. If two sandboxes contain customization changes to the same artifact and both are being published, then the sandbox that is being published last is given an option (by the sandbox manager) to overwrite the changes for that artifact from the sandbox that was published first. If the user working in the second sandbox decides to force the publication of the second sandbox, then the changes published by the first sandbox are overwritten. These types of conflicts can also occur with shared metadata files such as resource bundles that store translatable strings.

If there is a concurrent change made in the mainline code after the sandbox was created and the user attempts to publish that sandbox, such conflicts are detected at publication time and errors messages occur.

2.2.1.3 Guidelines for One Sandbox, Multiple Users

Regardless of whether the sandbox is intended for publishing or for test-only, if you want to permit multiple users to work in a single sandbox simultaneously, follow these guidelines:

- Multiple concurrent users in the same sandbox must operate only on different and unrelated objects.

  For example, if `user1` updates `object1`, then `user2` can update `object2` but should not update `object1`. Be aware that if both modifications involve changes to translatable strings, then saving changes to separate objects around the same time may still cause a conflict in the resource bundle that stores the translatable strings.

- If multiple users update the same artifact concurrently (either the same object or the same underlying frequently modified file), then they will get a concurrent update error. In this case, the second user's changes will not be saved (the **Save** button will be disabled) and one of the users will have to cancel and try again.

  CRM Application Composer retains any uncommitted changes in the UI. For example, if the user is editing a groovy expression when the error is encountered, then the expression is retained in the editor so the user can copy and paste the expression to try customizing again. (No metadata corruption or partial transactional commits will happen.)

- Users in the same sandbox will see the changes created by one another. The latest version of each object gets loaded on-demand the first time it is viewed in CRM Application Composer. Users can refresh their view to the latest definition from MDS and see the changes listed in the tree structure at the left-hand side of the CRM Application Composer UI. However, if there are ADF Business Components customizations, then users might need to log out and log back in again to see those changes reflected in the UI.

2.2.1.4 Guidelines for Multiple Sandboxes, Multiple Users

If you want to permit multiple users to work in multiple sandboxes, follow the guidelines in Section 2.2.1.3, "Guidelines for One Sandbox, Multiple Users," and these guidelines:

- There can be any number of test-only sandboxes operating concurrently. That is, multiple users can use multiple sandboxes concurrently for testing if these sandboxes will never be published. Sandboxes that are used for testing only, and that are not published, cause no conflicts with each other. Be aware, however, that all modifications will be lost when the sandboxes are destroyed.

  Note: Even though sandboxes that are for test-only cause no conflicts with each other because they are not published, multiple users who work in the same test-only sandbox still must follow the guidelines in Section 2.2.1.3, "Guidelines for One Sandbox, Multiple Users."

- For sandboxes that are not for test-only and will be published, you can have multiple concurrent sandboxes only if they operate on mutually exclusive artifacts. For example, you can have one sandbox that contains a page that is being customized to add a task flow, and another sandbox that contains a different page from a different application.

  Note: For CRM Application Composer, use only one sandbox at a time that is intended for publishing. You cannot have multiple concurrent sandboxes that are intended for publishing. You can still have other sandboxes that are test-only.

- If an artifact is updated in both the mainline code and in the sandbox (or in two different sandboxes), when the sandbox is published, such conflicts are detected and an error is raised. At this point, administrators should cancel the publishing of the sandbox to avoid overwriting previous changes.

Note: For a sandbox that contains ADF Business Components customizations, log out and log back in again after switching in or out of this sandbox to avoid any inconsistencies between the runtime caches and the ADF Business Components definitions.

Log out and log back in when you are performing any of the following sandbox-related actions:

- Before entering a sandbox
- After exiting a sandbox
- After publishing a sandbox
- After destroying a sandbox

2.2.2 Setting Up Sandboxes

When you create a sandbox, the currently available metadata is gathered into a sandbox session. You designate a sandbox to be the active sandbox by choosing either the newly created sandbox or selecting from a list of existing sandboxes in a table. The active sandbox is the context for all changes. After you activate the sandbox, you can make customization changes to the application artifacts and these changes will be stored in the sandbox. The sandbox uses a database to store the actual change information. After you are satisfied with the changes, you can publish the sandbox, or deploy the flexfield, and the changes will be merged into the mainline code and the sandbox will be archived.

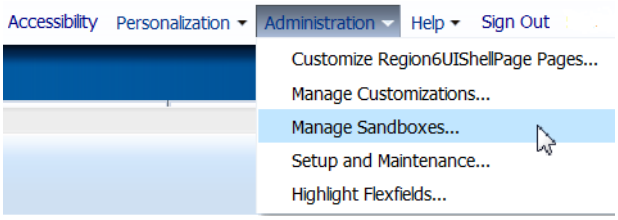
To set up a sandbox:

1. Access the sandbox manager.

   For non-flexfield sandboxes, you can access the sandbox manager by selecting the **Manage Sandboxes** menu item from the **Administration** menu in the global area of Oracle Fusion Applications.

   Figure 2–5 shows the Manage Sandboxes menu selection from the Administration dropdown list.

Figure 2–5 Accessing the Sandbox Manager

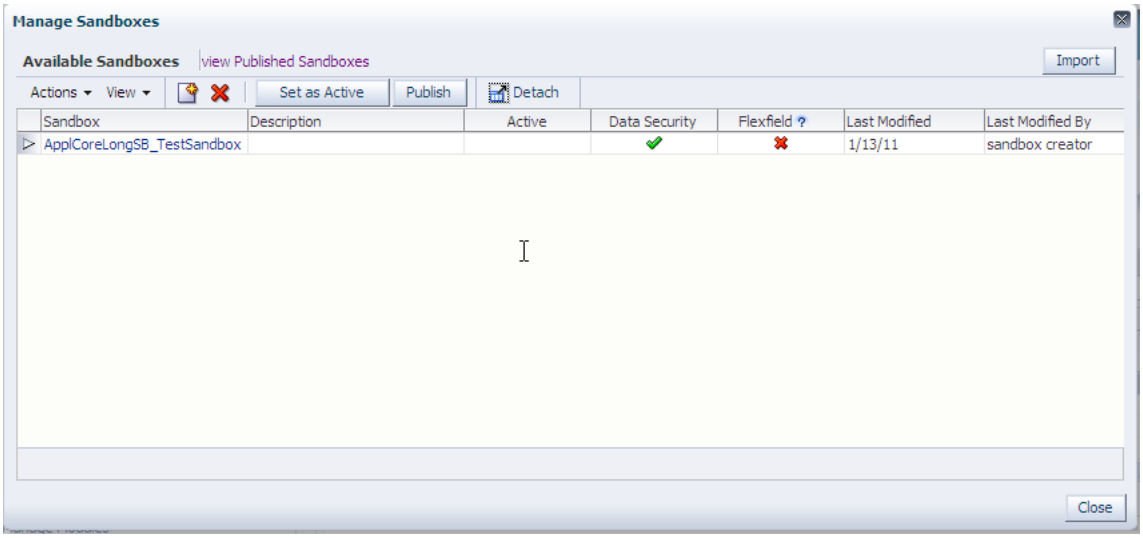


   For flexfields, you must be in the Manage Descriptive Flexfields task or the Manage Extensible Flexfields task and then select the **Deploy Flexfield to Sandbox** menu item to deploy the customizations to a flexfield sandbox. Flexfield changes are stored in flexfield metadata in the mainline database and are available to users only when the flexfield is deployed, as described in Section 5.7, "Deploying Flexfield Configurations." Thus for flexfields, the remaining tasks in this procedure do not apply.

2. In the Manage Sandboxes dialog, you can create a new sandbox, select from a list of sandboxes in the Available Sandboxes page, import a sandbox as a file, or

perform other sandbox functions. Figure 2–6 shows a Manage Sandbox dialog with one available sandbox.

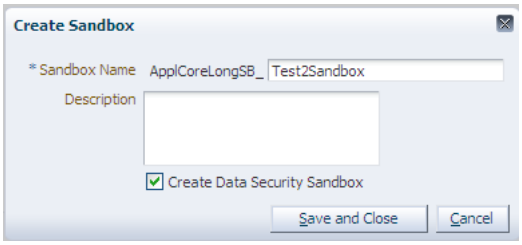
Figure 2–6 Manage Sandboxes Dialog



3. Create a new sandbox by clicking the **New** icon or selecting **Actions** then **New**
4. In the Create Sandbox dialog, enter a name for the sandbox. If you want a security-enabled sandbox, then select **Create Data Security Sandbox** and click **Save and Close**. Figure 2–7 shows the dialog with security enabled.

> **Note:** Because setting up the security sandbox requires duplicating the schema for Oracle Fusion Data Security tables, this will always be a lengthy operation in CRM Application Composer. Allow sufficient time for the process to be completed and do not to terminate it early. You may want to defer customizing security and enabling the security sandbox until you are sure that customizations are required.

Figure 2–7 Create Sandbox Dialog

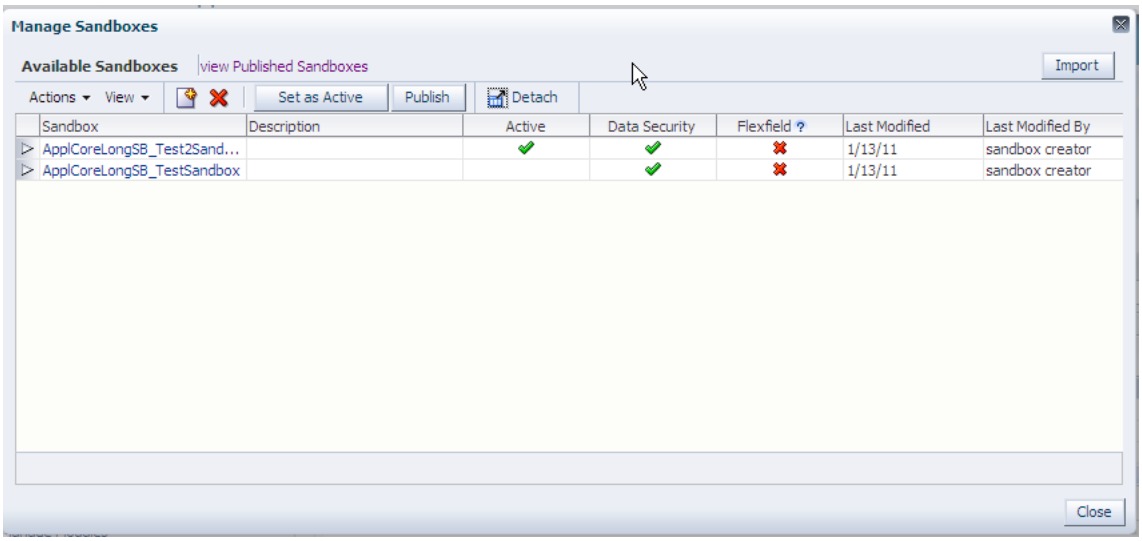


You will see a confirmation dialog when the sandbox has been successfully created. Click **OK** to dismiss the dialog.

5. Identify the active sandbox. You can make the newly created sandbox or an existing sandbox the active sandbox.

   In the Manage Sandboxes dialog, in the Available Sandboxes page, select the sandbox you want to make active and click **Set as Active** as shown in Figure 2–8.

Figure 2–8 Manage Sandboxes Dialog



Only one sandbox can be active at one time. The customization changes are captured in the active sandbox.

6. Export a sandbox.

   A sandbox can be exported as a file for transporting, sharing, and other uses where packaging it as a file is required. Consequently, a sandbox exported as a file can be imported.

   In the Sandbox Details dialog, click **Download All**. Enter the location and file name for the exported sandbox.

7. Import a sandbox.

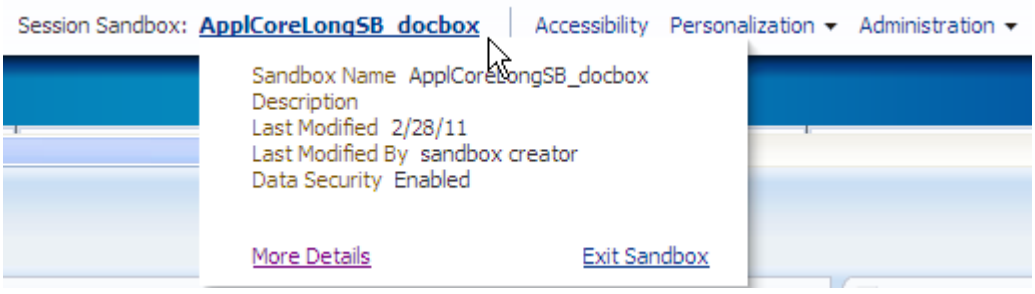
   In the Manage Sandboxes dialog, click **Import**.

   In the Import dialog, click **Browse** to navigate to the file or enter the fully qualified file name.

8. Exit the sandbox.

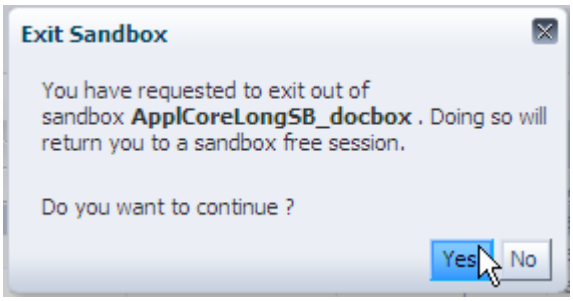
   Mouse over the sandbox name next to the **Session Sandbox** label in the global area of the Oracle Fusion applications to open a popup dialog, as shown in Figure 2–9. Click **Exit Sandbox**.

Figure 2–9 Sandbox Popup Dialog



In the Exit Sandbox dialog, click **Yes**, as shown in Figure 2–10.

Figure 2–10 Exit Sandbox Dialog



2.2.3 Publishing Sandboxes

If there are changes to the mainline code from another source and you publish your sandbox, then the mainline code is not overwritten. If there are conflicts, then you will be warned and given a chance to fix the conflicts.

To publish a sandbox:

1. Make the customization changes to the application by going to the various customization environments. For example, you can create new objects and customize the objects in CRM Application Composer.

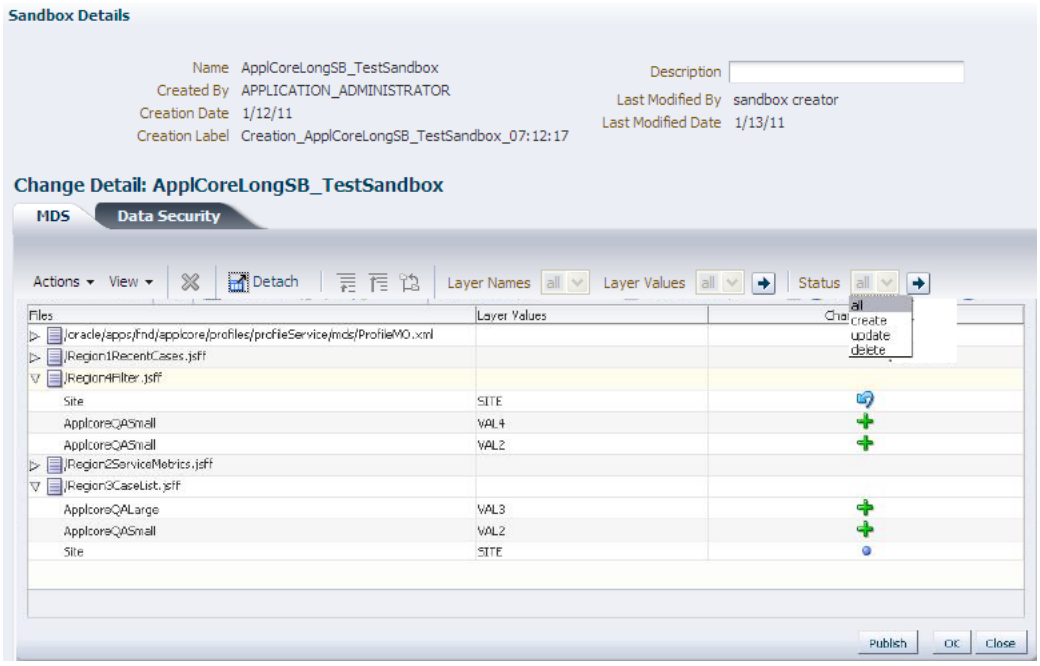
   You can make metadata changes in the sandbox. Metadata changes are usually associated with MDS. These include making changes to a page, to Oracle ADF customization, and to Oracle ADF business objects. For more information about making changes to the page, see Chapter 3, "Customizing Existing Pages." For more information about making changes to business objects, see Chapter 4, "Customizing Objects."

   You can make security changes by applying them to a security-enabled sandbox. You can test your security changes and policies before you publish the sandbox to commit the changes. For more information about customizing security policies, see Section 9.2, "Defining Security Policies for Custom Business Objects." For more information about custom business objects, see Chapter 15, "Customizing Security for ADF Application Artifacts."

2. Test or validate the changes in runtime using test or production systems and any combination of validation setups.

   In the Manage Sandboxes dialog, click on the sandbox link to launch the Sandbox Details dialog. You can view the layers and objects where you have made customization changes, as shown in Figure 2–11.

Figure 2–11 Sandbox Details Dialog



3. Publish the sandbox to the mainline.

   After you are satisfied with the changes, select the sandbox and click **Publish** in the Manage Sandbox dialog or in the Sandbox Detail dialog to commit the changes to the mainline.

2.3 Viewing and Diagnosing Runtime Customizations

You use the Manage Customizations dialog to view and diagnose runtime customizations that have been made to application pages.

By default, the Manage Customizations dialog displays the customizations that have been performed by the logged-in user. Administrators can optionally see the customizations made by other users.

> **Tip:** You can also use the Manage Customizations dialog to move customizations to another environment, for example from one test environment to another. For more information, see Section 2.4.1, "Downloading and Uploading Customization Files Using the Manage Customizations Dialog."

2.3.1 Before You Begin Using the Manage Customizations Dialog

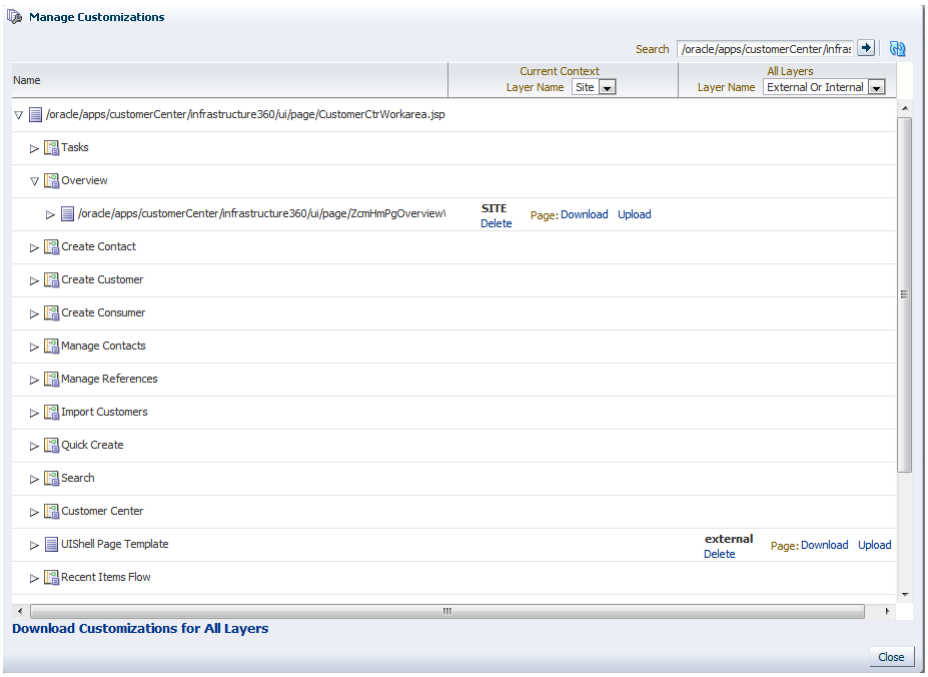
You need to do the following before you can use the Manage Customizations dialog:

- You need to have specific privileges to access the Manage Customizations dialog. Please contact your security administrator for details.
- To access the Manage Customizations dialog, choose **Manage Customizations** from the Administration menu in the global area.

2.3.2 Viewing Customizations Using the Manage Customizations Dialog

Figure 2–12 shows the Manage Customizations dialog with all artifacts related to the `CustomerCtrWorkarea.jsp` file.

Figure 2–12 Manage Customizations Dialog



As shown in Figure 2–12, a customization was made to a page fragment file in the Site customization layer, and a customization was made to the UIShell Page Template in the external layer. By default, the customizations in the **Current Context** column show the customizations that the currently logged-in user would see in a running version of the application. All customizations, including those the currently logged-in user might not see (perhaps because of security policies), are listed in the **All Layers** column.

For example, say you are a customization developer. Because you as a user have been assigned a particular role, the page that you see will display only those customizations appropriate for your role. Those customizations will be listed in the **Current Context** column (provided you chose the proper layer from the **Layer Name** dropdown menu). This helps you to determine what customizations are affecting what you are seeing. There may, however, be customizations that you do not see. You can view those customizations from the **All Layers** column.

Sometimes, an administrator might need to view personalizations that were made by other users. For example, a user might have made an error while personalizing a page and that page no longer displays for the user. Because the user cannot display the page, the user cannot access the page to correct the error. In this case, the administrator can access the page, request to see the user's changes, and delete those changes to restore the page to its original settings.

To view customizations:

1. Go to the page for which you wish to view customizations.
2. From the **Administration** menu in the global area of Oracle Fusion Applications, choose **Manage Customizations**.

Tip: After you are in the Manage Customizations dialog, you can change the page for which you are viewing customizations using the **Search** field.

3. From the **Layer Name** dropdown list for the **Current Context** column, select the customization layer for which you want to see the customizations as the user (or users) see it. For more information about customization layers, see Section 1.2, "Understanding Customization Layers."
4. If you have administrator privileges and you want to see the customizations for a user other than yourself, then select **Select User** from the **Layer Name** dropdown list for the **All Layers** column, and select the user. You can select multiple users by repeating this step.

2.3.3 Backing Out Customizations

Metadata labels identify the state of the objects in an MDS repository at a given point in time. Labels can serve as save points to which you can roll back your customizations if the customizations create problems. You roll back to a label by *promoting* that label *to the tip*.

To create a label across the entire MDS repository, use the `createMetadataLabel` Oracle WebLogic Scripting Tool (WLST) command as described in the "Creating Metadata Labels" section of the *Oracle Fusion Middleware Administrator's Guide*. You can also create a label when you save a customization in Page Composer.

To roll back all customizations, use the `promoteMetadataLabel` WLST command as described in the "Promoting Metadata Labels" section of the *Oracle Fusion Middleware Administrator's Guide*.

To roll back only the customizations for a specific page, use the Manage Customizations dialog accessed from Page Composer as described in the following steps. Note that when you use the Manage Customizations dialog, you are rolling back only the customizations for the page and its `pageDef` file. You are not rolling back the the other customizations made at the label's save point.

Note: Some labels are created automatically. For example, customizations that are created in a sandbox are automatically labeled when the sandbox is published. You can identify an automatically-created label by its prefix. For a list of these prefixes, see the "Managing Oracle Fusion Applications-Specific Labels in the Oracle Metadata Repository" section in the *Oracle Fusion Applications Administrator's Guide*.

To promote a page's customizations to the tip:

1. Go to the page for which you wish to view customizations.
2. From the **Administration** menu in the global area of Oracle Fusion Applications, choose **Customize *page_name* Pages** to open the page in Page Composer.
3. In the tool bar, click **Manage Customizations**.
4. To promote a customization to the tip, click **Promote** for the corresponding artifact.
5. In the Promote Documents dialog, select the label that you want to promote to the tip and click OK.

2.4 Downloading and Uploading Customization Files

As explained in Section 1.2, "Understanding Customization Layers," customizations are stored in an XML file. You can download the customizations into an XML file on your local machine and you can upload that file into other environments. You may need to download or upload a customization file for the following reasons:

- You need to diagnose issues seen in the test environment.
- You need to send files to Oracle Support Services for further diagnosing.
- You want to import a customization into another environment. For example, a customization developer using JDeveloper might need to see customizations done by someone else.
- You need to migrate a customization into a production environment, such as the production environment.

There are three ways you can download and upload customization files.

- Using the Manage Customizations dialog.
- Using Oracle WebLogic Scripting Tool (WLST) commands.
- Using Oracle Enterprise Manager Fusion Applications Control.

When working with a specific customization, use the Manage Customizations dialog. When working with a set of customizations, use either the WLST commands or Fusion Applications Control. When migrating from one environment to another, such as from the full test environment to production, use Fusion Applications Control.

2.4.1 Downloading and Uploading Customization Files Using the Manage Customizations Dialog

The Manage Customizations dialog enables you to upload and download customization files for a given page. Use the Manage Customizations dialog to upload or download minor customizations that affect only the customization file, such as disabling or hiding a field. For example, dropping a data control on the page affects the CPX file and changing text on the page affects resource bundles. If there is a possibility that the customization affected other files, use the WLST commands as described in Section 2.4.2, "Downloading and Uploading Customization Files Using WLST Commands" or use Fusion Applications Control as described in Section 2.4.3, "Downloading and Uploading Customization Files Using Fusion Applications Control."

If you manually edit the customization file before uploading it, ensure that the syntax of the edited contents is correct. Otherwise the uploading might produce incorrect instructions.

To download and upload customizations using the Manage Customizations dialog:

1. Go to the page for which you wish to download or upload customizations.
2. From the **Administration** menu in the global area of Oracle Fusion Applications, choose **Manage Customizations**.

> **Tip:** After you are in the Manage Customizations dialog, you can change the page for which you are viewing customizations using the **Search** field.

3. To download a file, click the **Download** link for the corresponding customization. The file will be downloaded to your local machine.
4. To upload a file, click the **Upload** link for the corresponding customization. In the Upload Customization dialog, click **Choose File** and select the appropriate file. Click **OK** to upload the file.

> **Note:** The name of the file you upload must be the same as the name of the file you are replacing.

5. To download all customizations of the page for all layers, click the **Download Customizations for All Layers** link, located at the bottom of the window. This will download the `AllCustomization.zip` file, which contains all the customization XML files for the page.

2.4.2 Downloading and Uploading Customization Files Using WLST Commands

You can use WLST commands to upload and download customization files for one page, multiple pages, the navigator menu, and Oracle Application Development Framework (ADF) Business Components.

> **Note:** You can also use Fusion Applications Control to upload and download sets of customizations, as described in Section 2.4.3, "Downloading and Uploading Customization Files Using Fusion Applications Control."

Example 2–1 shows how to export all customizations for a specific application.

Example 2–1 WLST Command to Export All Customizations for an Application

```
exportMetadata (application='application name',
server='server name', docs='/**',
excludeBaseDocs='true',
toLocation='temp location')
```

Example 2–2 shows how to export the site level customizations of the navigator menu.

Example 2–2 WLST Command to Export Site Level Customizations of the Navigator Menu

```
exportMetadata (application='application name',
server='server name', docs='oracle/apps/menu/mdssys/Site/SITE/root_menu.xml.xml',
toLocation='temp location')
```

For more information about uploading and downloading customization files, see Section 10.2.4, "Importing Customizations into Your Workspace." For information about uploading and downloading the customizations of resources, see Section 16.2, "Translating Resource Bundles from Metadata Services Metadata Repository."

For more information about the `exportMetadata` command, see the "Managing the Metadata Repository" chapter in the *Oracle Fusion Middleware Administrator's Guide*.

2.4.3 Downloading and Uploading Customization Files Using Fusion Applications Control

Use Fusion Applications Control to migrate customizations from one environment to another, such as from the full test environment to production. You can also use Fusion Applications Control to download and upload a set of customizations. For more information, see the "Transferring Metadata Using Fusion Middleware Control" section of the *Oracle Fusion Middleware Administrator's Guide*. The referenced procedure describes using Fusion Middleware Control, but also applies to Fusion Applications Control.

Part II

Business User Customizations and Extensions

Part II contains the following chapters:

3

Customizing Existing Pages

This chapter describes how administrative users can customize pages in Oracle Fusion applications at runtime using Page Composer.

This chapter includes the following sections:

3.1 About Customizing Existing Pages

The pages in Oracle Fusion applications provide content and functionality to users that enable them to complete their tasks (for example, learning about a product or service, keeping up with sales data, or submitting an order) as easily and efficiently as possible. Because different users have different needs, Oracle Fusion applications enable you to customize pages to fit those needs using Page Composer. End users can use Page Composer to *personalize* pages. For example, end users can reorganize content in dashboards to place the content they use most frequently at the top of the page. Administrative users can use Page Composer to *customize* pages in any layer of the application or to customize the *UI shell template* used by CRM applications—the template used for the base UI for all CRM application pages. For example, administrative users can add a logo or contact information to the header and footer of the application.

Most of what you do in Page Composer is perform actions on the objects, or *components*, on a page. Components are grouped together on the page into boxes to simplify management of related components. For example, you might group components relating to a particular project or to a particular subject. The grouped components are referred to as *child* components. Page Composer enables you to edit an individual child component or manage all the grouped child components at one time by editing the box containing the child components. For example, you might want all the related child components to have the same style (background color, font, and such) so that users can more easily see their relationship.

Figure 3–1 shows an application page as the user sees it. The components in the red outline are grouped together. Figure 3–2 shows the same application page in Page Composer, where you can see the dashed outline of the box containing the grouped components.

Figure 3–1 Components in an application page

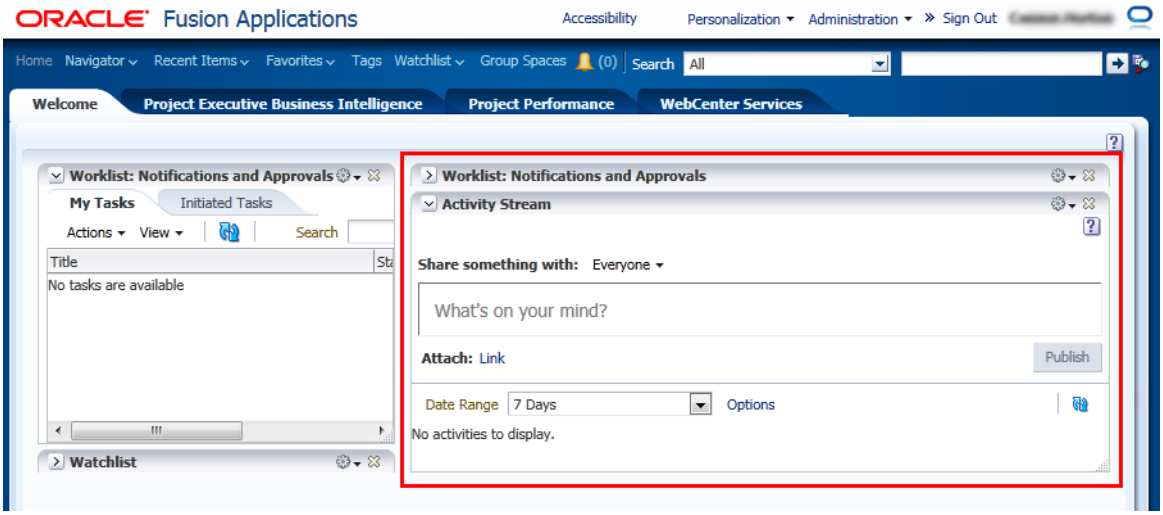
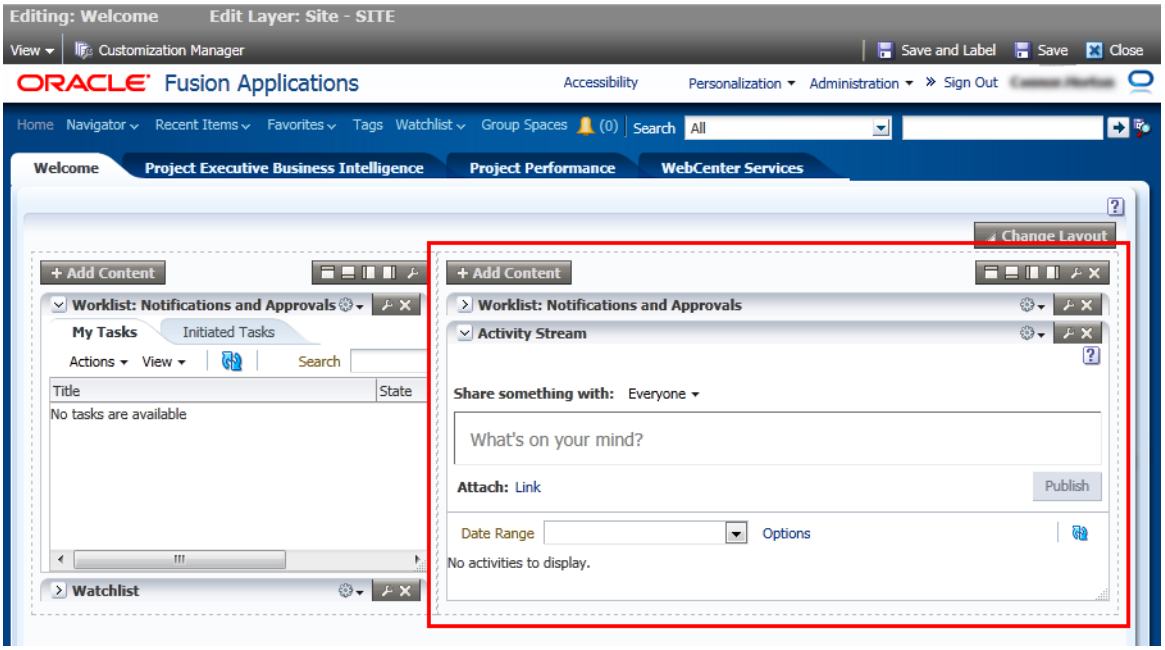


Figure 3–2 Components in Page Composer

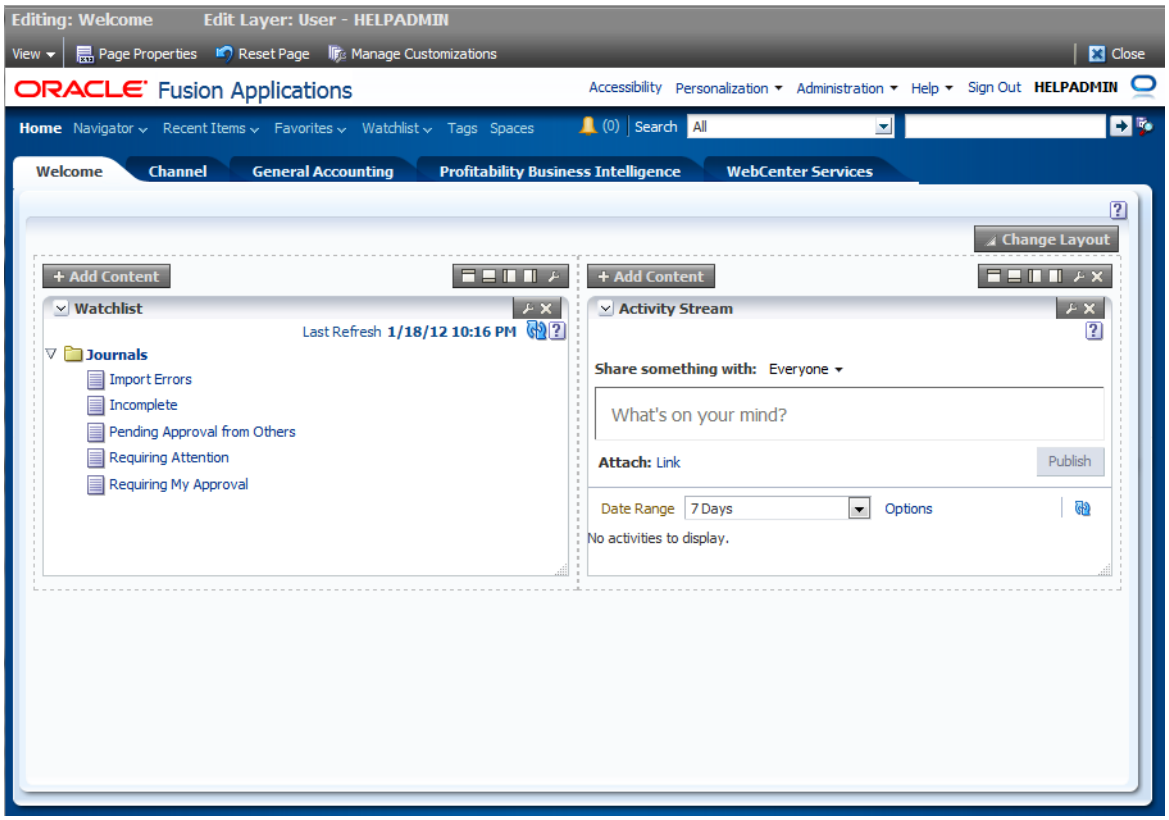


3.1.1 Page Composer User Interface Overview

Page Composer provides two views for working with page content:

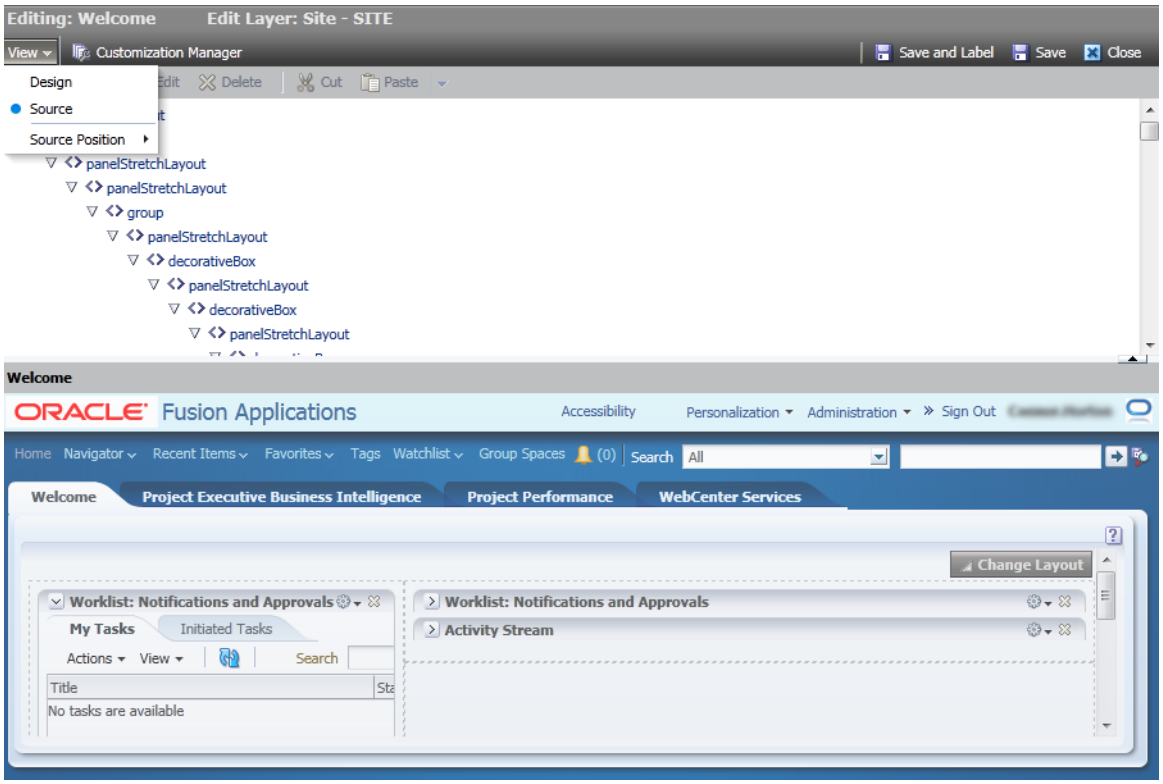
- Design view (Figure 3–3) provides a WYSIWYG rendering of the page and its content, where controls are directly selectable on each component. Design view is available in CRM and non-CRM applications.

Figure 3–3 Page Composer's Design view



- Source view (Figure 3–4) provides a combined WYSIWYG and hierarchical rendering of page components for advanced users (such as developers or users with knowledge of ADF artifacts, JavaScript, and Expression Language (EL)). Source view is available for non-CRM applications and when editing the UI Shell template in CRM applications.

Figure 3–4 Page Composer's Source view



For more information, refer to the "Introducing Design View and Source View" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

3.1.2 Effects of Editing Objects That Display on Multiple Pages

Sometimes objects are included on multiple pages. The effect of changing such an object depends on how the object is included on the page and whether you are working with a CRM or non-CRM application.

- If you use Page Composer to make changes to an object that is displayed on multiple pages (and is not part of a shared task flow), only the object on the page you edit is affected.
- If the object is part of a shared task flow, the change will affect the object on all pages that include the shared task flow, with one exception. If the shared task flow includes embedded logic that makes use of data from the page, the embedded logic might override the changes you make in Page Composer.
- For CRM applications, you can change objects globally (on all pages that display the object) by editing them through CRM Application Composer, as described in Section 4.2, "Editing Objects."

3.1.3 What You Can Do with Pages at Runtime

Depending on your role, the type of page, and the type of application, you can perform different actions in Page Composer.

3.1.3.1 Non-Dashboard Pages in CRM Applications

Sales Administrators, Marketing Operations Managers, and Channel Managers can customize non-dashboard pages in the following ways:

- Add task flows, portlets, documents, layout components, and other objects to a page

  Refer to Task: Add Components to a Page.

- Change a page layout

  Refer to Task: Change the Layout of a Page.

- Move objects on a page

  Refer to Task: Move Components on a Page.

- Change the display of the object using the tools in the object header (for example, expand/collapse an object, hide/show/reorder columns in a table)

3.1.3.2 Dashboard Pages in CRM Applications

Users can personalize their dashboard pages and administrators can customize site, internal/external, and job-role dashboard pages with the tasks mentioned in Section 3.1.3.1, "Non-Dashboard Pages in CRM Applications" and in the following additional ways:

- Show/hide objects on a page

  Refer to Task: Show and Hide Components on a Page.

- Remove objects on from a page

  Refer to Task: Delete Components from a Page.

3.1.3.3 Pages in Non-CRM Applications

Administrators can customize pages in non-CRM applications with the tasks mentioned in Section 3.1.3.1, "Non-Dashboard Pages in CRM Applications," Section 3.1.3.2, "Dashboard Pages in CRM Applications," and in the following additional ways:

- Customize a page title

  Refer to Task: Customize a Page Title.

- Customize a tasklist menu

  Refer to Task: Customize a Tasklist Menu.

- Customize attributes for a flexfield on a page

  Refer to Task: Customize Attributes for a Flexfield on a Page.

> **Notes:** Many pages in Oracle Fusion applications include *flexfields*. Flexfields allow you to add custom attributes to a page. You add flexfields to a page using Manage Descriptive Flexfields and Manage Extensible Flexfields, then customize the attributes through Page Composer. For more information about adding flexfields to a page, refer to Chapter 5, "Using Flexfields for Custom Attributes."

- Customize popup content

Refer to Task: Customize Popup Content.

- Provide values for the properties associated with pages and the objects the pages contain

  Refer to Section 3.3, "Editing Component Properties in Page Composer."

- Edit the component header and other display options

  Refer to Task: Edit Component Header and Other Display Options.

- Edit the look and feel of the component

  Refer to Task: Edit Component and Content Style.

- Edit component's unique parameters

  Refer to Task: Edit Component Parameters.

- Roll back changes to a page or task flow

  Refer to Task: Reset a Page or Task Flow to a Previously Saved Version.

- Wire pages, task flows, and portlets to each other

  Refer to Task: Allow Certain Component Property Values to be Persisted Across Sessions.

3.1.3.4 UI Shell Template in CRM Applications

CRM application administrators can customize the UI Shell template used by all pages in the application. Refer to Section 3.4, "Editing the UI Shell Template Used by All Pages."

3.1.4 What You Cannot Do with Pages at Runtime

Business users cannot perform the following tasks in a runtime environment with Page Composer:

- Make a page personalizable

  To make a page editable by end users, a developer must use Oracle JDeveloper to enable personalization. For more information, refer to Section 17.2, "Allowing Pages to be Personalized by End Users in Page Composer."

- Edit user interface text that is part of a skin

  To edit user interface text that is part of a skin, a developer must use JDeveloper to change the resource bundle used by the skin. For more information, refer to the "How to Apply Skins to Text" section in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

- Edit text that is part of the embedded help on the page

  To edit text that is part of the embedded help on a page, a developer must use JDeveloper. For more information, refer to Section 18.4, "Customizing or Adding Static Instructions, In-field Notes, and Terminology Definitions."

- Change ADF taskflows

  To edit ADF taskflows, a developer must use JDeveloper. For more information, refer to Chapter 11, "Customizing and Extending ADF Application Artifacts."

- Change ADF Business Components objects

  To edit the ADF Business Components objects (for example, to add validation to an ADF Business Components object), a developer must use JDeveloper. For more

information, refer to Chapter 11, "Customizing and Extending ADF Application Artifacts."

- Change the UI shell template in a non-CRM application

  For applications other than CRM, a developer must use JDeveloper to modify the UI shell template. For more information, refer to Section 11.10, "Editing the UI Shell Template."

- Change CRM administrative pages (such as Set Up and Look Up Management)

  To edit the CRM administrative pages, a developer must use JDeveloper.

- Change an object globally (change an object on every page on which it is displayed)

  For CRM applications, you can change objects globally by editing them through CRM Application Composer, as described in Section 4.2, "Editing Objects."

- Add a custom attribute to a page using the flexfield feature

  If a flexfield exists on a page, you must use the appropriate manage flexfield task to add the custom attributes to the page before you can work with them in Page Composer. For more information, refer to Chapter 5, "Using Flexfields for Custom Attributes."

- Change mobile web pages (pages built using Trinidad components for mobile clients)

  To edit mobile web pages, a developer must use JDeveloper. For more information, refer to Section 11.4, "Editing Pages."

3.1.5 Before You Begin Customizing Existing Pages

Before you implement customizations in applications, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with runtime customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can begin customizing existing pages:

- Confirm the page is customizable.

  For CRM applications:

  - Any user can personalize Dashboard pages.
  - Sales Administrator, Marketing Operations Manager, and Channel Manager users can customize transactional pages (such as Edit).

  For non-CRM applications you can only customize pages if a developer has enabled customization for the page.

- Optionally, set up a sandbox.

  Page Composer can use sandboxes to manage your customizations. For more information, refer to Section 2.2, "Using the Sandbox Manager."

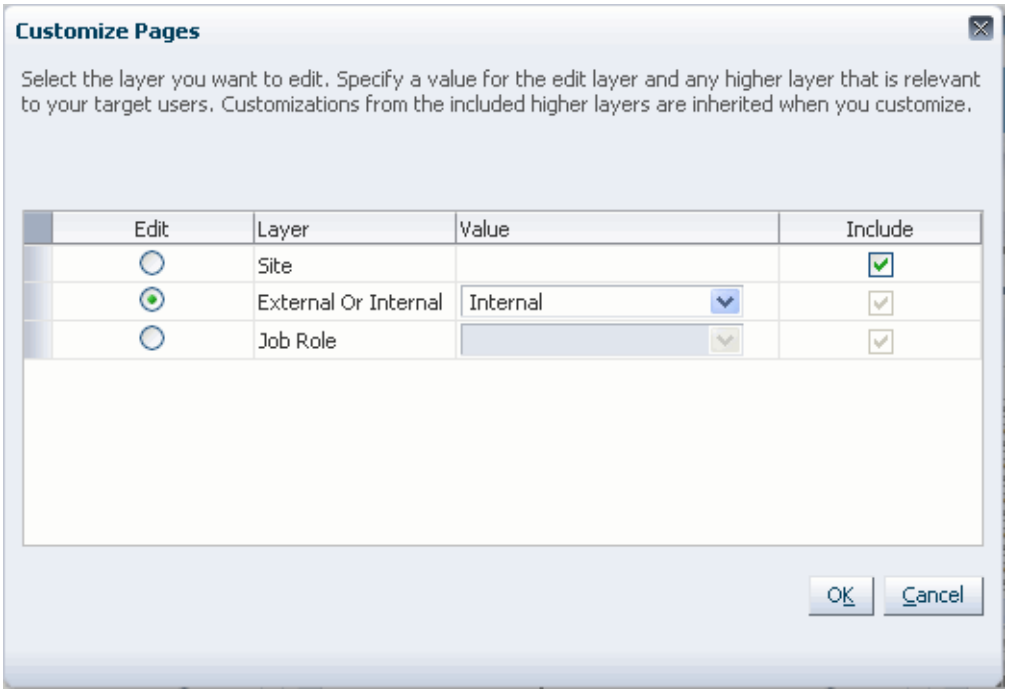
- Access Page Composer.

  - To access Page Composer, navigate to the page you want to edit, then, from the **Administration** menu in the global area of Oracle Fusion applications, choose **Customize** ***page_name*** **Pages**.

  - To customize existing pages, you will need the correct privileges. Contact your security administrator for details.

- If you have more than one layer available for customization, when you open Page Composer, the UI displays the Layer Picker dialog (Figure 3–5), which you use to specify the layer that you want to edit and its customization context. In the **Edit** column, select the layer you want to edit. The layers that are selected in the **Include** column will inherit any changes you make to the layer you edit.

Figure 3–5 Layer Picker Dialog



For more information on customization layers, including selecting a layer and customization context, refer to Section 1.2, "Understanding Customization Layers."

3.2 Editing a Page in Page Composer

This section describes how to perform basic editorial tasks, such as editing page components, changing the layout of the page, and the like. The tasks available to you depend on your role, the type of page, and the type of application. For details, refer to Section 3.1.3, "What You Can Do with Pages at Runtime."

> **Note:** If the object you want to edit is included on multiple pages, refer to Section 3.1.2, "Effects of Editing Objects That Display on Multiple Pages," before editing the object.

Task: Add Components to a Page

To add components to a page, in Design view, click **Add Content** at the top of the container component to which you want to add the new component.

For non-CRM pages, you can also add components in Source view (where UI widgets are available). To add components in Source view, you select the container component, then, in the Source view toolbar, click **Add Content**.

In either mode (Design view or Source view), clicking **Add Content** opens the Resource Catalog, enabling you to select from a wide range of task flows, portlets, layout components, and other types of resources. For more information on adding components to a page, refer to the "Adding a Component to a Page" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

> **Note:** If you want to display custom attributes, you may be able to use a flexfield that has been defined for the page. Flexfields are available for many pages in Oracle Fusion applications, except for the CRM applications. For more information, refer to Chapter 5, "Using Flexfields for Custom Attributes."

Task: Change the Layout of a Page

Page layout defines the number, placement, and orientation of content regions on a page. A page's initial layout style is selected when the page is created. Some style selections can be switched even after you have added content to the page. Other style selections, notably the Blog and Web Page page styles, do not support layout changes after creation.

To change the layout of a page, in Design view, click **Change Layout**. For more information, refer to the "Changing Everyone's Page Layout" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Move Components on a Page

To move components on a page, in Design view, drag and drop the component.

For non-CRM pages, you can also cut and paste or drag and drop components in Source view, or you can access the Component Properties for the container component and rearrange the components on the **Child Components** tab. For more information, refer to the "Rearranging Child Components" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Show and Hide Components on a Page

> **Note:** You can perform this task only on dashboard pages in CRM applications, on non-dashboard pages in CRM applications if you have the proper job role, or in non-CRM applications.

You can control whether to show or hide a component on a page. For example, you have a list of checkboxes, and if a user selects checkbox B, you want a button to display. You could set the Show Component property on the button to be an EL expression that says #{if checkboxB.selected = true}, meaning that if the selected value of checkbox B is "selected" then display the button.

There are two ways to hide a component—hide the individual component or hide the box containing the component.

To hide an individual component, click the Edit icon in the component header, click the **Display Options** tab, uncheck the **Show Component** box, and click **OK**. For more information, refer to the "Working with Component Display Options" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*. You can instead, click the Edit icon on the containing box's toolbar, then, on the **Child Components** tab, uncheck the box next to the component you want to hide, and click **OK**. For more information,

refer to the "Hiding and Showing Child Components" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

To hide the box and all its child components, click the Edit icon in the box's header, click the **Display Options** tab, uncheck the **Show Component** box, and click **OK**. For more information, refer to the "Working with Component Display Options" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Delete Components from a Page

Note: You can perform this task only on dashboard pages in CRM applications, on non-dashboard pages in CRM applications if you have the proper job role, or in non-CRM applications.

WARNING: Only delete a component if you are positive that no other components or processes are dependent on the component you delete. We strongly suggest you instead hide a component if you are unsure.

If you are sure no other components or processes are dependent on a component, you can delete the component from a page by clicking the Delete icon in the component header.

Note: Some components might not be able to be deleted, such as mandatory or indexed fields or components that are installed as part of the Oracle Fusion applications.

If you delete a box, all of the child components—any components contained in the box—are also deleted. For more information, refer to the "Deleting Layout Components" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Customize a Page Title

Note: You can perform this task only in non-CRM applications.

The page title is specified in three places:

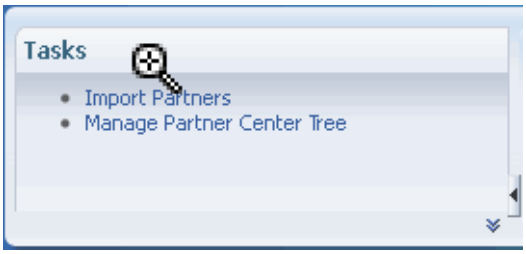
- The Page Title property in the Task List Properties tab in Page Composer, which affects the browser title
- The Label property in the Task List Task Properties tab in Page Composer, which affects the tasks lists menu entry, page heading, and tab title
- The Label property for the navigator menu item in the Manage Menu Customizations task, which affects the navigator menu entry

You must change all three entries to change the page title. To change the first two entries in Page Composer, follow the steps below. To change the navigator menu entry, refer to Chapter 6, "Customizing the Navigator Menu."

1. If Source view is not already displayed, switch to Source view. From the **View** menu, select **Source**.

2. In the Design region (at the bottom of the page), click the task list as shown in Figure 3–6.

Figure 3–6 Selecting a Task List in the Design Region



3. When asked to confirm that you want to edit the task flow, click **Edit**.

   This selects the task list entry in the Source region.

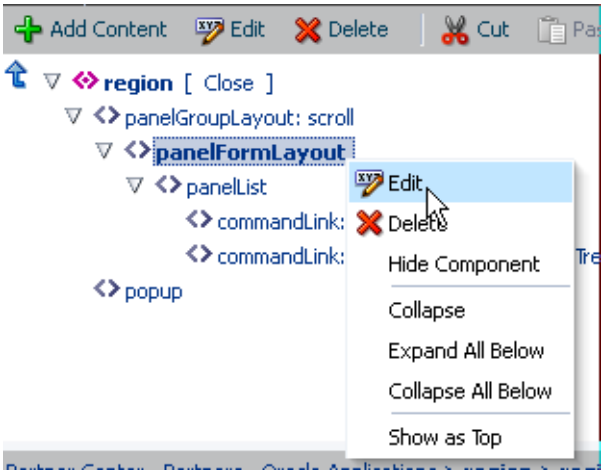
4. In the Source region, click Edit Task Flow next to the task list as shown in Figure 3–7.

Figure 3–7 Opening the Task Flow Editor

5. Again, when asked to confirm that you want to edit the task flow, click **Edit**.

6. In the Source region, right-click the `panelFormLayout` node, and select **Edit** as shown in Figure 3–8.

Figure 3–8 Editing the panelFormLayout Node

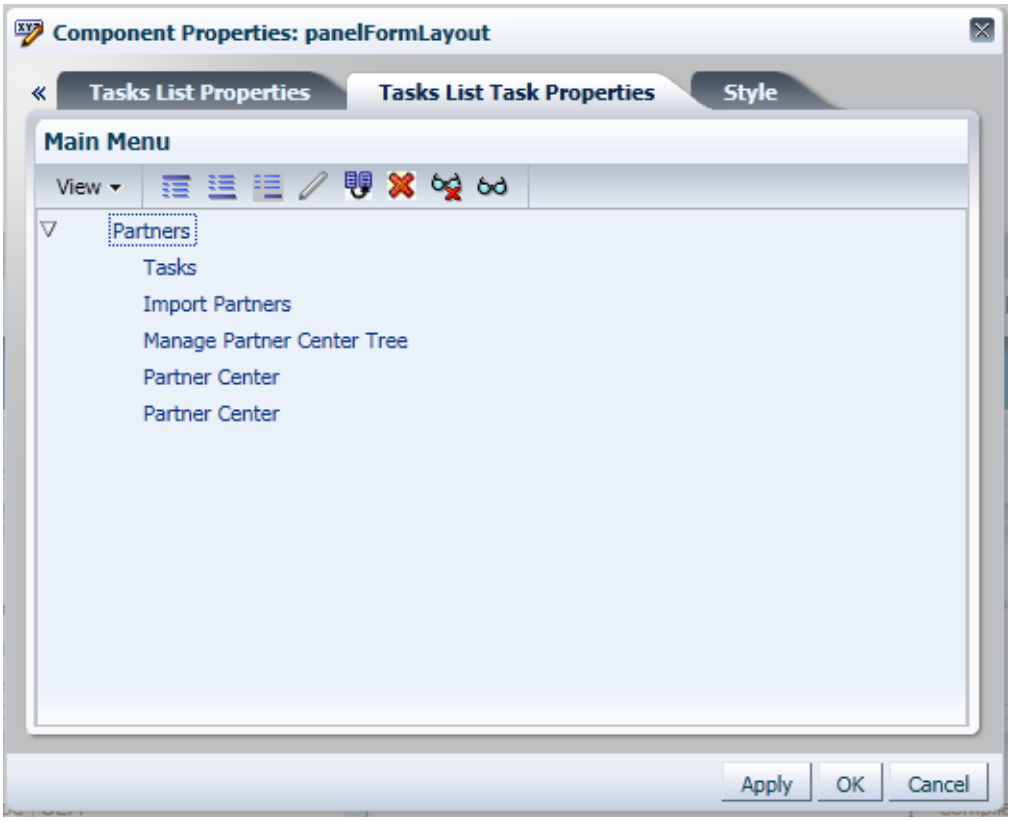


   The Component Properties dialog for `panelFormLayout` is displayed.

7. On the Tasks List Properties tab, in the **Page Title** box, type the title for the page, then click **OK** to save your changes and close the Component Properties dialog.

8. Click the **Tasks List Task Properties** tab.
9. Expand the tree to display the items in the tasklist as shown in Figure 3–9.

Figure 3–9 Component Properties — Tasks List Task Properties Tasklist Items



10. In the tree hierarchy, click the first child item ("Tasks" in Figure 3–9), then click the Edit icon in the toolbar.
11. In the **Label** box, type the title for the page, then click **OK** to save your changes and close the Component Properties dialog.

Task: Customize a Tasklist Menu

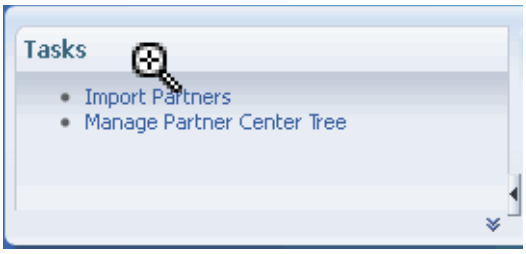
> **Note:** You can perform this task only in non-CRM applications.

Task lists enable you to provide links to task flows in your application or Web pages outside your application. For example, you can add links to frequently used task flows, so that users can quickly perform the most common tasks.

> **Notes:** Task lists can be customized only at the site level.

1. If Source view is not already displayed, switch to Source view. From the **View** menu, select **Source**.
2. In the Design region (at the bottom of the page), click the task list as shown in Figure 3–10.

Figure 3–10 Selecting a Task List in the Design Region



3. When asked to confirm that you want to edit the task flow, click **Edit**.

   This selects the task list entry in the Source region.

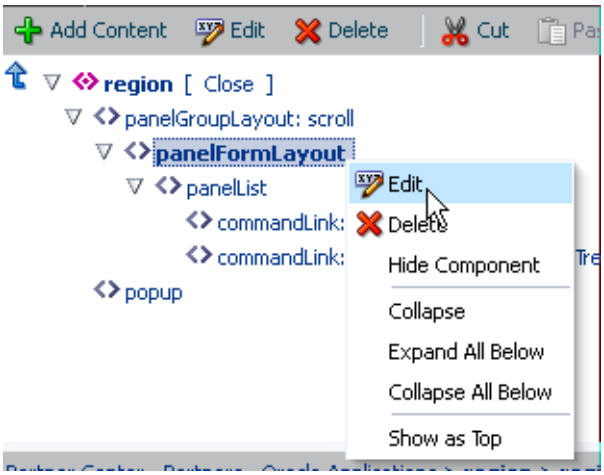
4. In the Source region, click Edit Task Flow next to the task list as shown in Figure 3–11.

Figure 3–11 Opening the Task Flow Editor

5. Again, when asked to confirm that you want to edit the task flow, click **Edit**.

6. In the Source region, right-click the `panelFormLayout` node, and select **Edit** as shown in Figure 3–12.

Figure 3–12 Editing the panelFormLayout Node

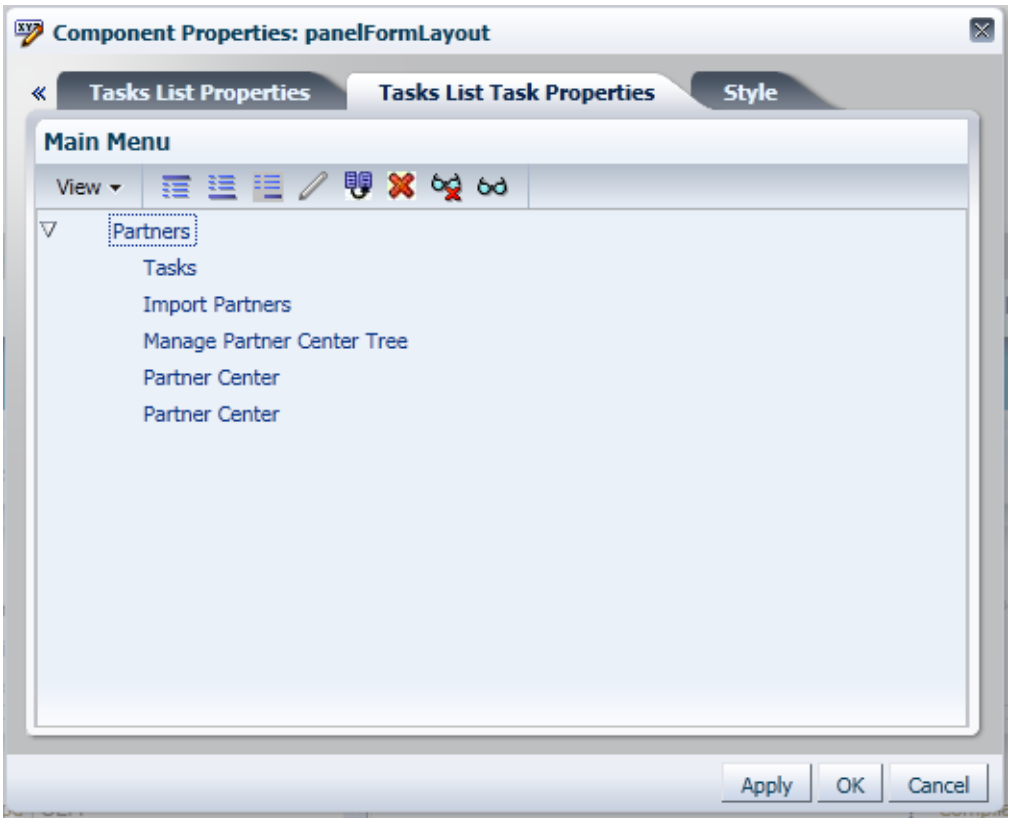


   The Component Properties dialog for `panelFormLayout` is displayed.

7. Click the **Tasks List Task Properties** tab.

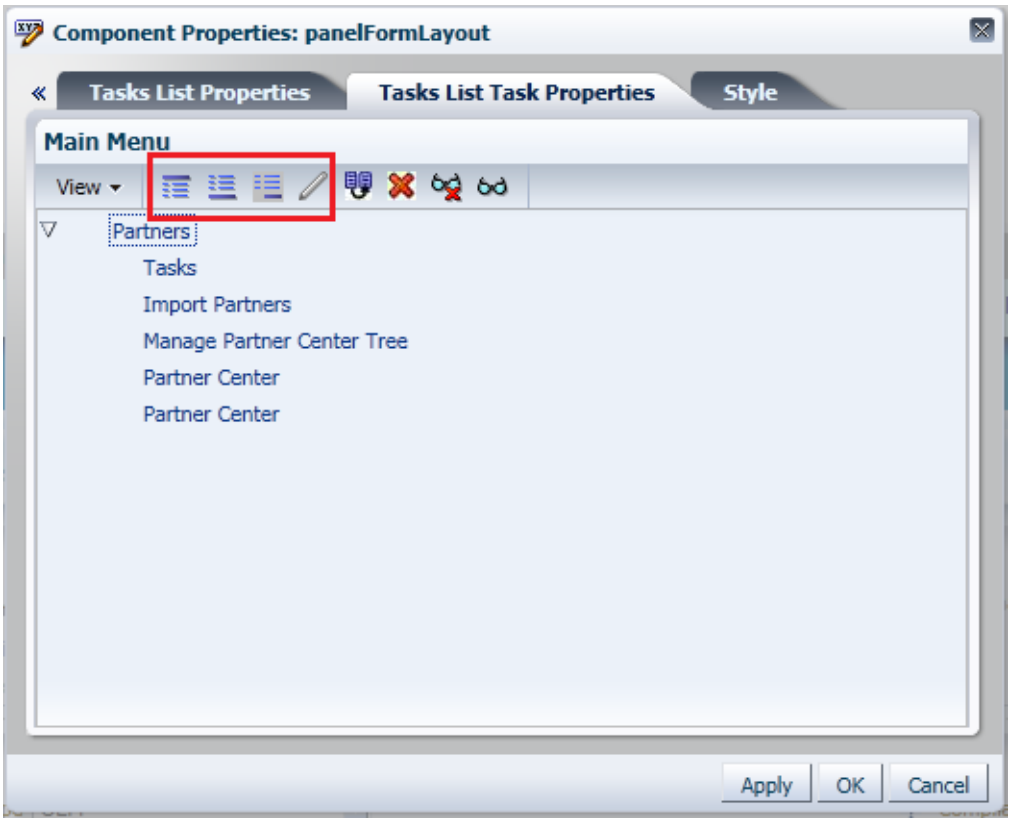
8. Expand the tree to display the items in the tasklist, as shown in Figure 3–13.

Figure 3–13 Component Properties — Tasks List Task Properties Tasklist Items



9. In the tree hierarchy, select an item and customize the tasklist by inserting a new item above, inserting a new item below, inserting a child item, or editing the current item by clicking the appropriate icon in the toolbar, as shown in Figure 3–14.

Figure 3–14 Component Properties — Tasks List Task Properties Toolbar



10. Enter or edit the following values:

- **Web Application**: Use the dropdown list to select the target web application.

  This list contains web applications that are defined in the deployments tables.

  Caution: If you enter a value for **Web Application** then you must enter a value for **Focus View Id**.

- **Focus View Id**: Enter the `focusViewId` of the target page. For example, */ServiceRequest*.

  Caution: If you enter values for **Web Application** and **Focus View Id**, do not enter a value for **Destination** and vice versa.

- **Action**: Enter the action that is taken when this item is selected by the user.

  Pages with actions are defined in the `adfc-config.xml` file, and these actions can navigate the user to a particular page. If you specify an action here, the **Web Application** and **Focus View Id** values are ignored. This **Action** attribute is used in an ADF Controller navigation.

- **Label**: Enter the label name for this new item. This is the name that appears on the tasklist menu. This label name also appears on the **Task** tab when opened if the page's `isDynamicTabNavigation="true"`.

- **Rendered**: Select to display the item in the tasklist. Uncheck to hide the item.

Note: When unchecked, the item is displayed in italics on the customization dialog. This allows you to identify items that are currently hidden from users.

- **Destination**: Enter the full URL for this item, such as *http://www.example.com*.

  The Destination attribute is used for navigation outside of the Oracle Fusion Middleware UI Shell pages. It opens in a new window.

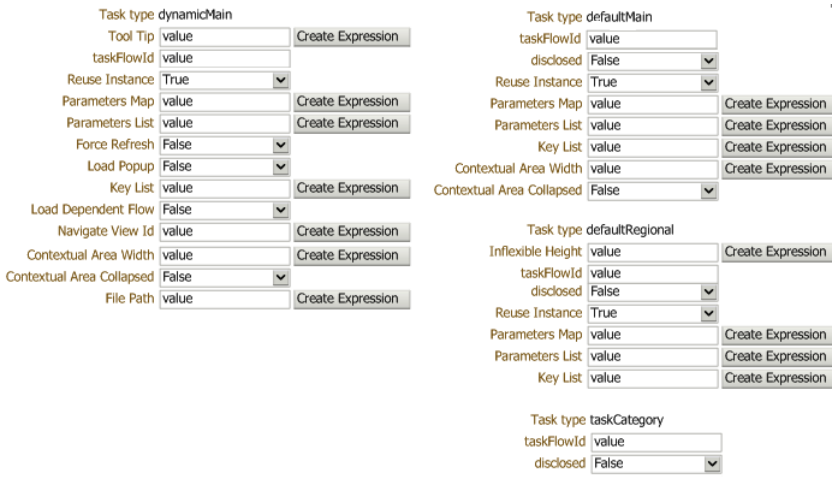
Note: Destination takes precedence over any specified Web Application

- **Task Type**: Choose the required task type for newly created items. Use the dropdown menu to select `dynamicMain`, `defaultMain`, `defaultRegional`, or `taskCategory`.

Caution: The task type can be specified by the administrator for the newly inserted item nodes only. It cannot be updated for an existing item node.

Figure 3–15 lists the properties that are applicable based on the task type of the currently edited item node.

Figure 3–15 Tasks List Task Properties — Task Types



11. Click **Apply** to save your changes, then click **OK** to save your changes and close the Component Properties dialog.

Task: Customize Attributes for a Flexfield on a Page

Note: You can perform this task only in non-CRM applications.

After you deploy an extensible or descriptive flexfield, you can use Page Composer to further control the custom attribute properties on a page-by-page basis. For example, you can hide some custom attributes or reorder how they appear on the page.

> **Note:** For information on flexfields, refer to Chapter 5, "Using Flexfields for Custom Attributes."

To customize flexfield values, edit the page in Source view. In Source view, navigate through the tree and expand the flexfield component (for example `descriptiveFlexfield`) to refer to the custom attributes. Click the flexfield component to display the Component Properties dialog box, where you can edit the values for the attributes.

Task: Customize Popup Content

> **Note:** You can perform this task only in non-CRM applications.

You can use Page Composer to customize the content in popup dialogs.

1. If Source view is not already displayed, switch to Source view. From the **View** menu, select **Source**.
2. Select the button that brings up the popup dialog.
3. Open the properties for the popup dialog.
4. Click the **Child Components** tab in the Component Properties dialog.
5. Edit the popup content.
6. Click **Apply** to save your changes, then click **OK** to save your changes and close the Component Properties dialog.

3.3 Editing Component Properties in Page Composer

> **Note:** You can perform these tasks only in non-CRM applications.

Components, such as task flows, portlets, documents, and layout components, carry with them a set of configurable properties that control the appearance and behavior of a particular component instance.

To edit component properties:

1. In Design view, click the Edit icon in the component header.
2. Click the appropriate tab in the Component Properties dialog.
3. Edit the properties.
4. Click **Apply** to save your changes, then click **OK** to save your changes and close the Component Properties dialog.

> **Note:** If the object you want to edit is included on multiple pages, refer to Section 3.1.2, "Effects of Editing Objects That Display on Multiple Pages," before editing the object.

For more information, refer to the "Setting Properties on Page Components" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Edit Component Header and Other Display Options

Typically, the **Display Options** tab presents settings that affect the display elements surrounding component content (that is, its chrome). Chrome includes the header, the **Actions** menu, **Expand** and **Collapse** icons, and the like. For example, use the display options on a task flow to hide or show a header, change the text in the header, enable or disable menus, show a tooltip for the component, and other options. Use the display options on an Image layout component to specify the image source URL and its optional link target. For more information, refer to the "Working with Component Display Options" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Edit Component and Content Style

Style and Content Style properties provide an opportunity to fine-tune your application look-and-feel at the component level. You can specify color, style, and margin settings on the selected component instance. For more information, refer to the "Working with Style and Content Style Properties" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Edit Component Parameters

Component parameters are settings, unique to the component type, that can control such things as the source of the component's content. Component parameters vary from component to component. For example, on some components they provide the opportunity to specify the source of task flow content; on other components they present read-only, application-generated identifiers that are used in maintaining a task flow instance's association with its customizations. For more information, refer to the "Working with Component Parameters" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

Task: Reset a Page or Task Flow to a Previously Saved Version

Page Composer provides controls for resetting a page or task flow to a previously-saved version or to its original out-of-the-box state.

The **Reset Page** button is available on the page in both Design view and Source view. For more information, refer to the "Reset Page" section in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

The **Reset Task Flow** button on the Source view toolbar is rendered only when editing a task flow. For more information, refer to the "Reset Task Flow" section in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

Task: Allow Certain Component Property Values to be Persisted Across Sessions

Certain attribute values of ADF Faces components can be persisted for end users. For example, on the column component, an end user can change the width of a column, and that width will still be in effect when the user logs back into the application. For information about which component properties can be persisted, refer to the "Introduction to Allowing User Customizations" section in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

To make a property persistable, list that property as a value for the persist parameter, using the procedures in the "Working with Style and Content Style Properties" section in *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*.

For more information about user personalization of components, refer to Section 17.3, "Configuring End User Personalization for Components."

3.4 Editing the UI Shell Template Used by All Pages

If you are customizing a CRM application, you can use Page Composer to edit the UI Shell template used by all pages in the application. If you are customizing a non-CRM application, you need to use JDeveloper. For more information, refer to Section 11.10, "Editing the UI Shell Template," or the "Introduction to Implementing the UI Shell" section in *Oracle Fusion Applications Developer's Guide*.

Note: You can also customize the Oracle Fusion Applications skin (for both CRM and non-CRM applications) as described in Chapter 19, "Customizing the Oracle Fusion Applications Skin."

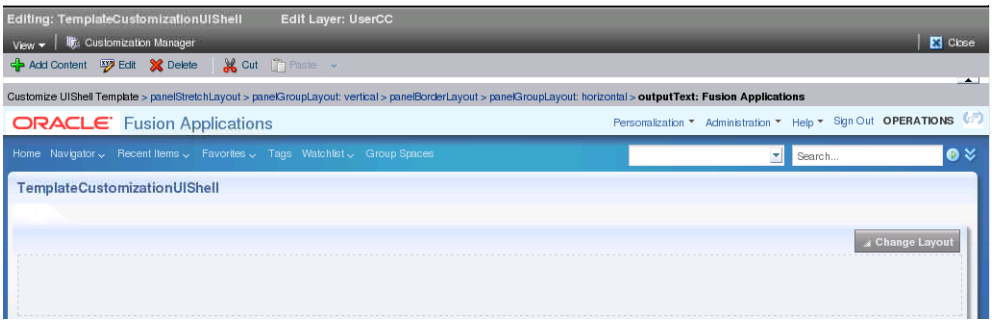
You edit the UI Shell template in Source view. (Source view is available in CRM applications only for the UI Shell template.) The only customizable parts are in the global area (as shown in Figure 3–16) and a footer panel. Components can be added to any of these areas using the task flow.

Figure 3–16 UI Shell Template — Global Area

Task: Customize the UI Shell Template

1. Open the UI Shell Template task flow, as shown in Figure 3–17.

Figure 3–17 Edit UI Shell Template



2. Add or edit components to the left or central portion of the global area (the header area that contains the logo). For example, you can brand your application with your company logo.

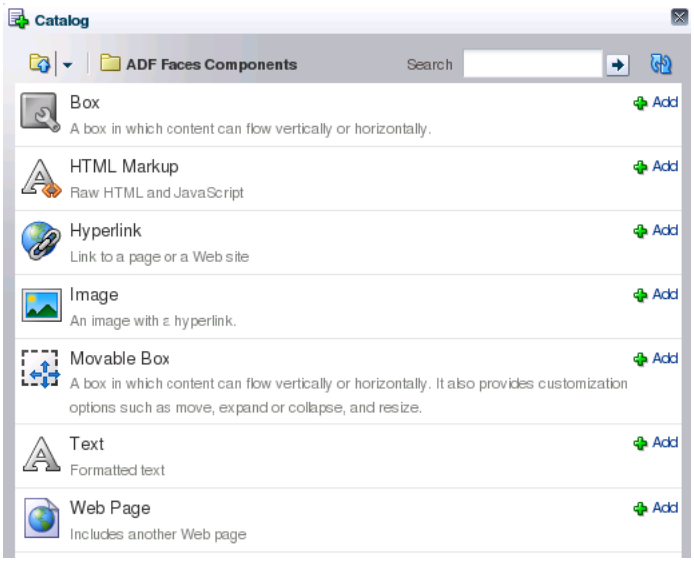
   You can also add a new component to the footer panel. For example, you can add company contact information. The footer panel is visible in edit mode even if no footer component has been added.

Note: If you need to do further customization, you can do with JDeveloper.

Task: Add a Component to the Global Area or the Footer

1. Select the portion of the global area to which you want to add a component, or select the footer, and click **Add Content** to open the component catalog.
2. Select **ADF Faces Components** to display the list of available components, as shown in Figure 3–18.

Figure 3–18 ADF Faces Components Catalog



3. Choose the component that you want to add and click the associated **Add** icon.

   The component is now displayed in the global area. You can now edit the component. For example, if you added the Text component, you can enter the text that you want displayed.

3.5 Editing Pages in Oracle JDeveloper After Using Page Composer

Using Page Composer, you can implement a variety of customizations on an application's pages. Pages that were created or customized in JDeveloper are further customizable in Page Composer, and page customizations that were implemented in Page Composer can be viewed in JDeveloper.

To view Page Composer customizations in JDeveloper, and potentially further customize the pages, you will need to export the customizations from the runtime environment and import them into the JDeveloper customization workspace. For more information, refer to the "Viewing ADF Library Runtime Customizations from Exported JARs" section in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

It is important to note that you cannot customize a given artifact at the same layer in both JDeveloper and Page Composer. You can, however, customize a given artifact in both tools provided the customizations are made at different layers. At run time, the tip layer customizations take precedence. For example, if you customize the label for a field in the `site` layer using Page Composer and customize the same label in the `global` layer using JDeveloper, the `site` layer customization will be displayed at run time.

4

Customizing Objects

This chapter describes how to use CRM Application Composer to customize and extend application artifacts in Oracle Fusion applications.

This chapter includes the following sections:

4.1 About Customizing and Extending Your Fusion Application with Objects

CRM Application Composer allows you to customize existing objects, attributes, and rules and create new ones for the following CRM applications:

- Sales
- Marketing
- Customer Center
- Trading Community Architecture
- Order Capture

If you want to customize applications other than those listed here, then you will need to use developer tools to create or change objects. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts." For more information about the other types of object customizations that you cannot perform in a runtime environment, see Section 4.1.2, "What You Cannot Customize in the Runtime Environment."

> **Tip:** If you want to customize objects in applications other than those listed here, then you might be able to use flexfields. For more information about flexfields, see Chapter 5, "Using Flexfields for Custom Attributes."

4.1.1 What You Can Customize and Create in the Runtime Environment

CRM Application Composer allows you to create and customize objects for the Sales, Marketing, Customer Center, Trading Community Architecture, and Order Capture applications. Objects are high level artifacts that typically manage data that resides in a corresponding database table.

Using CRM Application Composer, you can perform object customizations like the following:

- Edit existing objects.

  For example, you can edit the object's attributes or create custom attributes, add server script, create validation rules, create object workflows, and add saved searches.

- Edit the searches over your objects or create new ones.
- Create custom objects.

  When using CRM Application Composer to create an object, you do not need to create the corresponding table to store the data, the tool manages that for you.

- Extend existing work areas or create new work areas.

  You can edit pages for an existing object, or create pages for a custom object.

After you implement customizations on an object or create a custom object, you can use other tools to do the following:

- Add a custom object or a customized object to a report.

  For more information, see Chapter 8, "Customizing Reports and Analytics."

- Set security on custom objects.

  For more information, see Chapter 9, "Customizing Security for Custom Business Objects."

4.1.2 What You Cannot Customize in the Runtime Environment

While you can create objects using CRM Application Composer, the following are more advanced use cases for which you will need to use development tools:

- Customize or create an object contained in an application that is not Sales, Marketing, Customer Center, Trading Community Architecture, or Order Capture.

  In those cases, you must use JDeveloper instead of CRM Application Composer to implement customizations. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

- Edit relationships between preconfigured objects.

  You can use CRM Application Composer to create custom objects that are a child object or related object of an existing object. However, you cannot modify the relationships between preconfigured objects. To customize the relationships between preconfigured objects, you must use JDeveloper. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

- Create a new type of page for your new object. CRM Application Composer only allows you to create work area pages to expose the object in the application.

If you want to create a new type of page that does not fit the CRM Application Composer design pattern, you must use JDeveloper. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

- Edit the Navigator menu.

  To customize the navigator menu, you use the Manage Menu Customizations task in the Setup and Maintenance work area. For more information, see Chapter 6, "Customizing the Navigator Menu."

- Use managed beans to contain logic for a page.

  For example, if you want to add logic in response to a component event, you need to use JDeveloper. For more information, see the "Using a Managed Bean in a Fusion Web Application" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

4.1.3 Before You Begin Customizing and Extending Your Oracle Fusion Application with Objects

Before you customize or extend CRM applications using CRM Application Composer, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with runtime customizations and extensions, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

In addition, you will need to perform the following tasks before you can begin customizing or extending objects:

- Optionally, set up a sandbox.

  CRM Application Composer can use sandboxes to manage your customizations. For more information, see Section 2.2, "Using the Sandbox Manager."

- Launch CRM Application Composer.
  - You must have the necessary functional and data security privileges to access CRM Application Composer. Contact your security administrator for details.
  - You can enter the CRM Application Composer environment directly from the application you want to customize. To launch CRM Application Composer, log in as an administrator to the application you want to customize, and from the Navigator menu, choose **CRM Application Composer**.

- Select the Application to Customize.

  In addition to customizing the application you are logged in to, you can optionally implement customizations in objects that are common to multiple applications. To customize one of these common objects, select **CRM Common** from the **Application** dropdown list.

4.2 Editing Objects

You can customize objects in a number of ways using CRM Application Composer. Objects use metadata to store information about them, such as which attributes are displayed and how they are labeled. When you modify an object using CRM Application Composer, you are editing the object's metadata. For more information about how customizations are stored, see Chapter 1, "Customizing and Extending Oracle Fusion Applications." Also, for an overview of extensibility in CRM

Application Composer, see the "Extending CRM Applications: How It Works" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Changes that you make to an object appear on the associated pages that display the object, without requiring editing of the page. The only time you need to edit a page after customizing a object is if you add or remove an attribute in the object and want to add or remove that attribute on a page.

Before You Begin

If the object you want to edit is not displayed in the CRM Application Composer tree when you select the application you want to customize, then you cannot use CRM Application Composer. You will need to use JDeveloper, as described in Section 11.2, "Editing Existing Business Components."

When you want to edit an existing object, expand that object in the Objects panel to display the kinds of editors that are available for the object.

Task: Edit Attributes

You can change the properties of an attribute, such as changing the label or making it required. In CRM Application Composer, open the object you want to customize and use the Fields editor to make changes to the attribute. For more information, see the "Editing Fields: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

For custom attributes, you can configure all properties pertaining to the field type and the exposed properties in the taskflows. When you make changes to an existing attribute, these changes are reflected in all pages that display the object.

For preconfigured attributes on standard objects and the system-generated attributes on custom objects, the only properties you can configure in CRM Application Composer are Display Label and Hint Text. For other changes to these kinds of attributes, you will need to use JDeveloper, as described in Section 11.2, "Editing Existing Business Components."

Task: Add Attributes

You can use CRM Application Composer to add a custom attribute to an object using the Fields editor. Open the object you want to customize, and select **Fields**. For more information, see the "Editing an Object: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

If you add an attribute and want that attribute to appear on a page, you will also need to add it to the page. You can do this in CRM Application Composer using the Pages editor. For more information, see Section 4.3, "Editing a Page in CRM Application Composer."

Task: Add and Edit Business Rules

You can add validation rules, triggers, and object functions to an object in CRM Application Composer using the Server Scripts editor. For example, you can add snippets of Groovy script that are executed at specific points in the lifecycle (Create, Modify, Remove, BeforeInsert, and so on). Open the object you want to customize, and select **Server Scripts**. The Server Scripts editor contains an expression builder, which helps you compose the script for your business rule. For more information, see the "Groovy Scripting: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Add Buttons and Links

You can use CRM Application Composer to create buttons and links for an object. This can be accomplished from the Buttons and Links editor. Open the object you want to customize, and select **Buttons and Links**. For more information, see the "Buttons and Links: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Using CRM Application Composer, you define a button or link that can then be used on a page for the object. If you add a button or link and want it to appear on a page, you will also need to add it to the page. You can do this in CRM Application Composer using the Pages editor. For more information, see Section 4.3, "Editing a Page in CRM Application Composer."

Task: Edit the Web Service for the Object

You do not need to modify the web service for an object after adding or removing attributes. The payload is managed for you by CRM Application Composer. However, if you have client applications that use the service, you will need to regenerate the web service proxy with the new WSDL to allow access to new attributes. For more information, see the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Add Object Workflows

Object workflows connect changes in objects to subsequent actions, which allows you to automate your commonly used business processes. You can use CRM Application Composer to add an object workflow that conditionally responds to one of the following record modification events for the object:

- When a record is created
- When a record is updated
- When a record is deleted

In response to these events, you can choose to take one of the following actions:

- Field Updates — update a field
- Email Notification — send an email
- Task Creation — create a task
- Outbound Message — post a message
- Business Process Flow — initiate a business process flow

To add an object workflow, open the appropriate object, select **Object Workflows**, and click **New**. For more information, see the "Object Workflows: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

4.3 Editing a Page in CRM Application Composer

When you make changes to an object in CRM Application Composer, some kinds of changes are automatically reflected on the pages for an object. For example, a change to the label text for an attribute does not require you to edit the corresponding page. However, for the kinds of changes described in this section (for example, adding components and reordering fields), you will need to edit the pages that contain the object, in order for the change to appear.

The Pages editor in CRM Application Composer allows you to create and edit the three types of pages that are associated with an object.

- landing page

- creation page
- detail page

At runtime, these pages are displayed in the work area of the application. Each of these pages has specific, preconfigured behaviors that are commonly associated with the use of an object. For more information, see the "Creating a Work Area: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

CRM Application Composer allows administrative users to make specific types of changes to the pages that expose objects. The actions that you perform in CRM Application Composer follow a specific design pattern, and are limited to changes associated with the object:

- adding and removing attributes
- reordering attributes
- adding buttons and links
- adding subtabs and tree nodes

For other modifications to pages (such as layout or look and feel), you can use Page Composer, as described in Chapter 3, "Customizing Existing Pages."

Note: When you use CRM Application Composer to customize the properties of attributes and other components of an object, your customizations are applied to the object and are reflected in the customized application on the pages in which that object appears. There is no need to make subsequent modifications to the UI components that display them on the page. However, if you want to modify a property of a component on a particular page (for example, changing the tooltip for a field on a particular page), you can use Page Composer to make this kind of change. For more information about using Page Composer to edit the properties of UI components, see Section 3.3, "Editing Component Properties in Page Composer."

Before You Begin

In CRM Application Composer, you can create and edit only those pages that are associated with an object. Therefore you must start with an object. Whether you are customizing a preconfigured object or creating a custom object, the object must be created first.

Before editing the page for an object, you must launch CRM Application Composer, and select the application that contains the object you want to customize. For more information, see Section 4.1.3, "Before You Begin Customizing and Extending Your Oracle Fusion Application with Objects."

Task: Add Components to a Page

After you add a field or button to an object, you can use the Pages editor in CRM Application Composer to add the corresponding components to the object's pages. Open the appropriate object, select **Pages**, and then click on the page you want to edit. For more information about using CRM Application Composer to edit pages, see the "Editing Pages: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

To add other kinds of components to a page or to edit other types of pages, use Page Composer. For information about using Page Composer, see Section 4.3, "Editing a Page in CRM Application Composer."

Task: Remove Components from a Page

You can use the Pages editor in CRM Application Composer to remove components associated with an object (such as, buttons and fields) from the object's pages. Open the appropriate object, select **Pages**, and then click on the page you want to edit. For more information, see the "Editing Pages: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Reorder Fields on a Page

Using CRM Application Composer, you can change the order of fields on a page using the Pages editor. Open the appropriate object, select **Pages**, and then click on the page you want to edit. For more information, see the "Editing Pages: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Enable Instance-Level Grant Conveyance

When editing summary table on the landing page for a custom object, the Pages editor in CRM Application Composer provides the **Allow Access Grant** checkbox. Select this checkbox to allow the user of that page to give another user access to particular rows in that object. This option is also available for subtab tables. For more information, see Section 9.3, "Enabling End User Instance-Level Security Customization."

4.4 Creating Custom Objects

Objects are high level artifacts that manage data residing in a database table. When you create a custom object, the underlying infrastructure to store the data is managed by CRM Application Composer. You can also use CRM Application Composer to add validation, child objects, and the pages that will display your custom object.

Task: Create Custom Objects

You can add custom objects to your CRM applications that allow you to expose and capture additional data. To create a custom object in CRM Application Composer, select **Custom Objects** and click **New**. This launches a dialog that allows you to name and create the object. After creating the object, you can edit the details for the object. For more information, see the "Extending CRM Applications: How It Works" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Create Relationships Between Objects

With CRM Application Composer you can configure a custom object to be a related or child object of another object. To do this, click **Relationships** in the Common Setup panel. The Relationships editor allows you to define a relationship by selecting the source and target objects in the relationship.

You can also use CRM Application Composer to create custom objects that are a child object or related object of an existing object. To do this, select the existing object, and click **New** in the Child Objects or Related Objects area.

When you create an object as the child of another object, it will have cascading properties and it can only be used in the context of the parent object. It cannot be used as a standalone object or the child of another object. However, if you create a custom object and subsequently configure it as the child of another object, it can be used independently of the parent but it will not have cascading properties. For more

information, see the "Object Relationships: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Add Business Rules

You can add validation rules, triggers, and object functions to an object in CRM Application Composer using the Server Scripts editor. For example, you can add snippets of Groovy script that are executed at specific points in the lifecycle (Create, Modify, Remove, BeforeInsert, and so on). Open the object you want to customize, and select **Server Scripts**. The Server Scripts editor contains an expression builder, which helps you compose the script for your business rule. For more information, see the "Groovy Scripting: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Create Pages for the Object

After you have created an object, you can use CRM Application Composer to generate the pages (landing, creation, and detail) that expose the object in the work area of the application. You do this using the Pages editor. Open the object, and select **Pages**. For more information, see the "Creating a Work Area: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Add an Object Page to the Navigator menu

After you have created a custom object and the pages that display it, you might want to add one or more of those pages to the Navigator menu. For information about modifying the Navigator menu, see Chapter 6, "Customizing the Navigator Menu."

Task: Add Security for the Object

After you have created an object, you can use CRM Application Composer to implement security for the object. If you want to customize security for an object contained in an application that is not Sales, Marketing, or Customer Center, you will need to use Functional Setup Manager. For more information about customizing security, see Chapter 9, "Customizing Security for Custom Business Objects." For information about using CRM Application Composer to implement security, see the security chapter in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Add an Object to an Existing Report

The Custom Subject Areas editor in the Common Setup area of CRM Application Composer allows you create and customize subject areas that can be used in reports. You can use this editor to add objects, child objects, fields, date measuring, rollups, and aggregates to a subject area, as well as define which application roles can see them. To define a custom subject area, click **Custom Subject Areas** in the Common Setup panel, then click **New**.

After you define a custom subject area, a user of the application with the appropriate role, while running the BI Report Composer from the client, can select the report subject area and report on the objects and fields that were included as part of that subject area. For information about custom subject areas, see the "Editing an Object: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*. For more information about customizing reports, see Chapter 8, "Customizing Reports and Analytics."

4.5 Creating and Editing Search Objects

You can create and edit searches for an object in CRM Application Composer using the Saved Searches editor. Select an object and click **Saved Searches**. To edit an existing

search object, select the saved search you want to edit, and click the **Edit** icon. To create a new custom search object, click the **Create** icon. For more information, see the "Saved Searches: Explained" section in the *Oracle Fusion Applications CRM Extensibility Guide*.

In addition to saved searches, the work area for an object contains a regional search and the landing page contains a local search. Using the Pages editor, you can specify which attributes from the object are used in the search. Select an object and click **Pages**. Then click the search you want to edit, either **Edit Regional Search** or **Edit Local Search**. For more information, see the *Oracle Fusion Applications CRM Extensibility Guide*.

4.6 Editing Objects and Pages in Oracle JDeveloper After Using CRM Application Composer

Using CRM Application Composer, you can extend the application with custom objects, implement customizations on standard objects, and add and edit the pages that display those objects.

To see these customizations and custom objects in JDeveloper, you will need to export them from the runtime environment and import them into the JDeveloper customization workspace. For more information, see Section 10.2.4, "Importing Customizations into Your Workspace."

When using more than one tool to implement customizations and extensions, be aware of the following:

- Only standard objects and custom objects created in CRM Application Composer can be edited in CRM Application Composer.
- Only the pages associated with custom objects and customizable standard objects are editable in CRM Application Composer.
- Extensions and customizations implemented in JDeveloper are not editable in CRM Application Composer.
- Extensions and customizations implemented in CRM Application Composer are viewable in JDeveloper.

  For example, you can extend your application with a custom object created in CRM Application Composer, and then import it into the JDeveloper customization workspace. You can then use JDeveloper to create or customize a page or task flow to use the runtime-generated object.

> **Important:** Using JDeveloper to customize objects that are editable in CRM Application Composer is *not* recommended, because you cannot edit objects in CRM Application Composer that have been customized in JDeveloper. If you customize such an object in JDeveloper and subsequently open CRM Application Composer, the object will appear in the list of objects you can edit. But if you attempt to edit it, it can have an adverse impact on the application.

5

Using Flexfields for Custom Attributes

This chapter describes how to use descriptive and extensible flexfields to add additional attributes to your Oracle Fusion applications. You create these custom attributes using runtime tasks in the Setup and Maintenance work area.

This chapter includes the following sections:

5.1 About Using Flexfields

If you need to add company-specific attributes to a business object, such as size and color attributes for the product business object, then you can often use *flexfields* to add the desired custom attributes. A flexfield is a set of placeholder fields (*segments*) that is associated with a business object. Oracle Fusion Applications provides three types of flexfields — *descriptive*, *extensible*, and *key*. This chapter discusses how you can use descriptive and extensible flexfields to add additional attributes to Oracle Fusion applications. This chapter refers to these attributes as *custom* attributes.

> **Note:** This chapter does not discuss key flexfields, which you use to define keys such as part numbers, as explained in the product-specific documentation in Oracle Fusion Applications Help. In this chapter, the term *flexfield* applies to descriptive and extensible flexfields only.

Descriptive and extensible flexfields enable you to configure your applications to capture additional pieces of information (*attributes*) without having to perform custom development. The attributes that you add by configuring flexfields are available throughout the Oracle Fusion Middleware technology stack, allowing custom attributes to be used in user interface pages, incorporated into the service-oriented architecture (SOA) infrastructure, and integrated with Oracle Business Intelligence.

Note: For Sales, Marketing, Customer Center, Trading Community Architecture, and Order Capture applications, you use CRM Application Composer to add custom attributes instead of using descriptive and extensible flexfields. For more information, see Section 4.2, "Editing Objects."

5.1.1 What You Can Do with Flexfields at Runtime

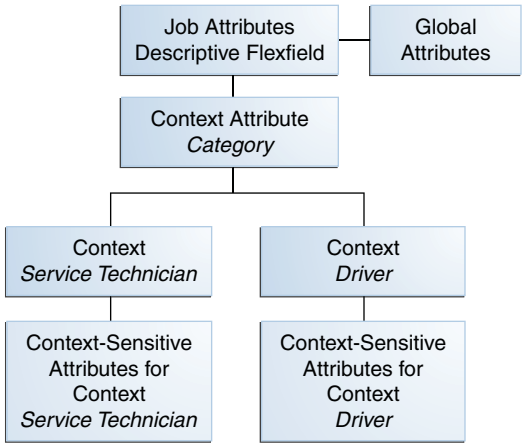
Many business objects in Oracle Fusion applications have an associated descriptive or extensible flexfield with which you can create custom attributes for the business object. A descriptive flexfield is the more basic of the two and is more commonly used. As explained later in this section, extensible flexfields offer more advanced features, such as hierarchical configurations.

The flexfield type determines how you configure the custom attributes.

- Descriptive flexfield configuration: A descriptive flexfield enables you to add three types of custom attributes to a page — *global*, *context*, and *context-sensitive*. Global attributes are always available as fields in the UI. Context-sensitive attributes appear on a UI page only under a certain condition or circumstance (the *context*).

  An example of where you would use context-sensitive attributes is for a job business object, as illustrated in Figure 5–1. Where some attributes are common to all jobs, such as whether the job is off site, some job attributes depend upon the job category. For example, you might want to store the service type for a service technician, and for a driver, you might want to store the required commercial driver's license (CDL) class.

Figure 5–1 Example of Descriptive Flexfield Global and Context-Sensitive Attributes



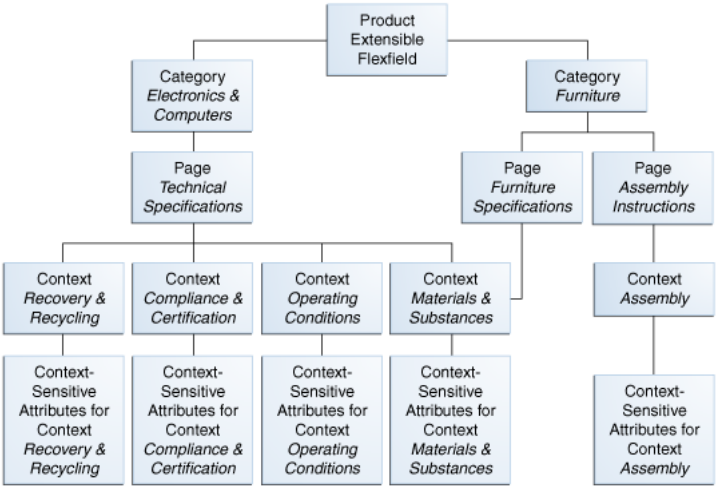
- Extensible flexfield configuration: An extensible flexfield enables you to add attributes to a page by configuring *categories*, *logical pages*, and *contexts*.

  Extensible flexfield contexts are somewhat different from descriptive flexfield contexts in that extensible flexfields enable you to configure multiple contexts, and you can group the contexts into categories. All extensible flexfields have at least one category (often referred to as the root category), and some extensible flexfields enable you to configure a hierarchy of categories, where a given category can inherit contexts from its parent categories. For more information about categories, see Task: Identify the Extensible Flexfield's Category Hierarchy Structure.

You use logical pages to arrange how the contexts appear in the user interface. For example, the extensible flexfield in Figure 5–2 has been configured to include a Technical Specifications logical page in the user interface for the Electronics and Computers category. The Technical Specifications logical page contains the attributes for four contexts — Recovery and Recycling, Compliance and Certification, Operating Conditions, and Materials and Substances. Figure 5–3 shows the user interface for the Technical Specifications logical page.

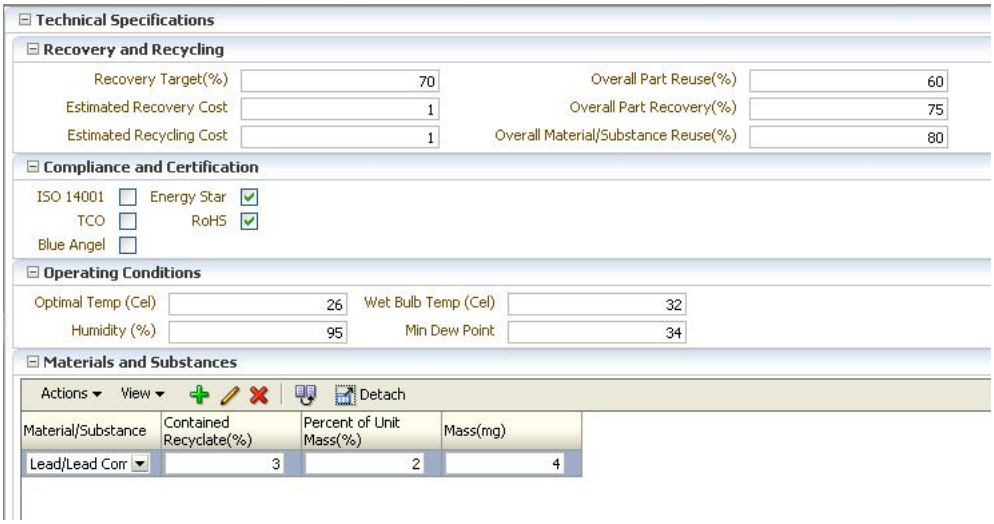
As shown in Figure 5–2, the user interface for the Furniture category has been configured to include a Furniture Specifications logical page and an Assembly Instructions logical page. Note that the two categories share the Materials and Substances context.

Figure 5–2 Example of Extensible Flexfield Categories, Pages, and Contexts



Another extensible flexfield feature is the ability to configure a context to store multiple rows per entity. For example, you can use a context to store all the materials and substances required to make a single product. The Materials and Substances context in Figure 5–3 has been configured for multiple rows, and is thus displayed as a table.

Figure 5–3 Technical Specifications Page with Associated Contexts



Use the Manage Descriptive Flexfields task, shown in Figure 5–4, and the Manage Extensible Flexfields task, shown in Figure 5–5, to create your custom attributes. In addition, some setup activities enable you to complete product-specific flexfield configuration. For example, you can use the Manage Item Classes task in the Product and Catalog Management application to manage a hierarchy of custom attributes for catalog items. For more information, see the product-specific documentation in Oracle Fusion Applications Help.

Figure 5–4 Manage Descriptive Flexfields Task

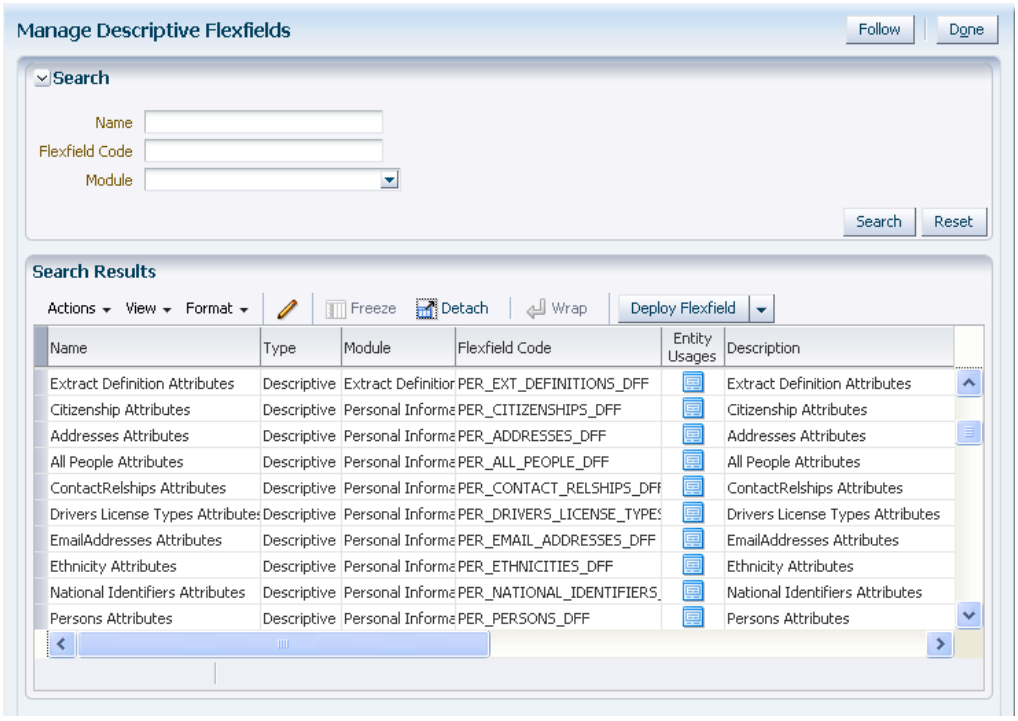
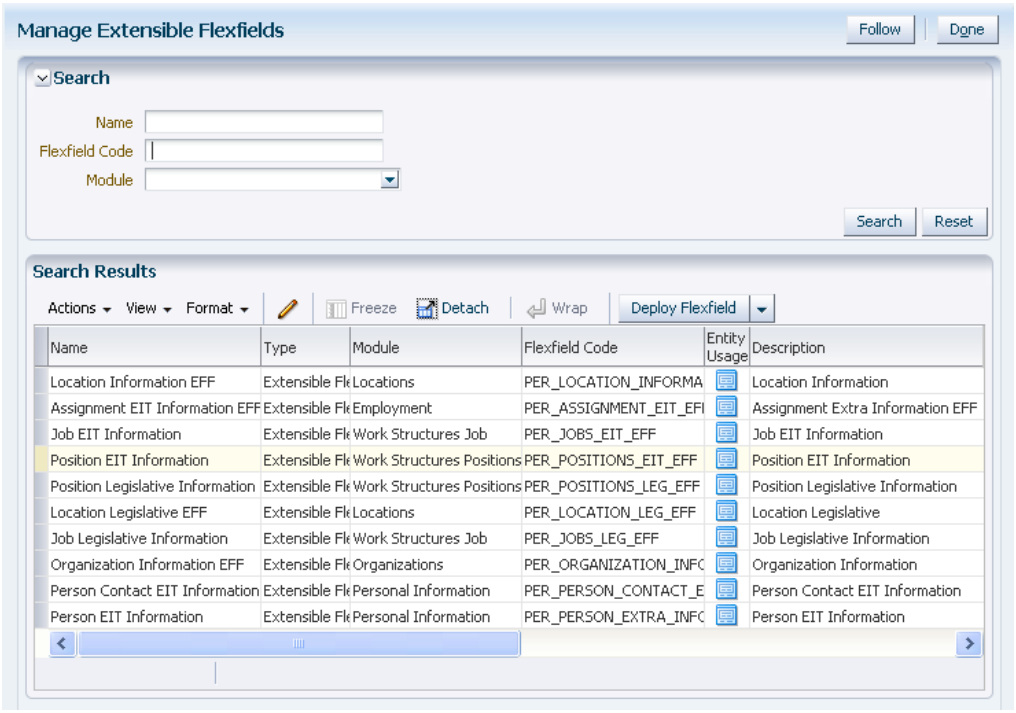
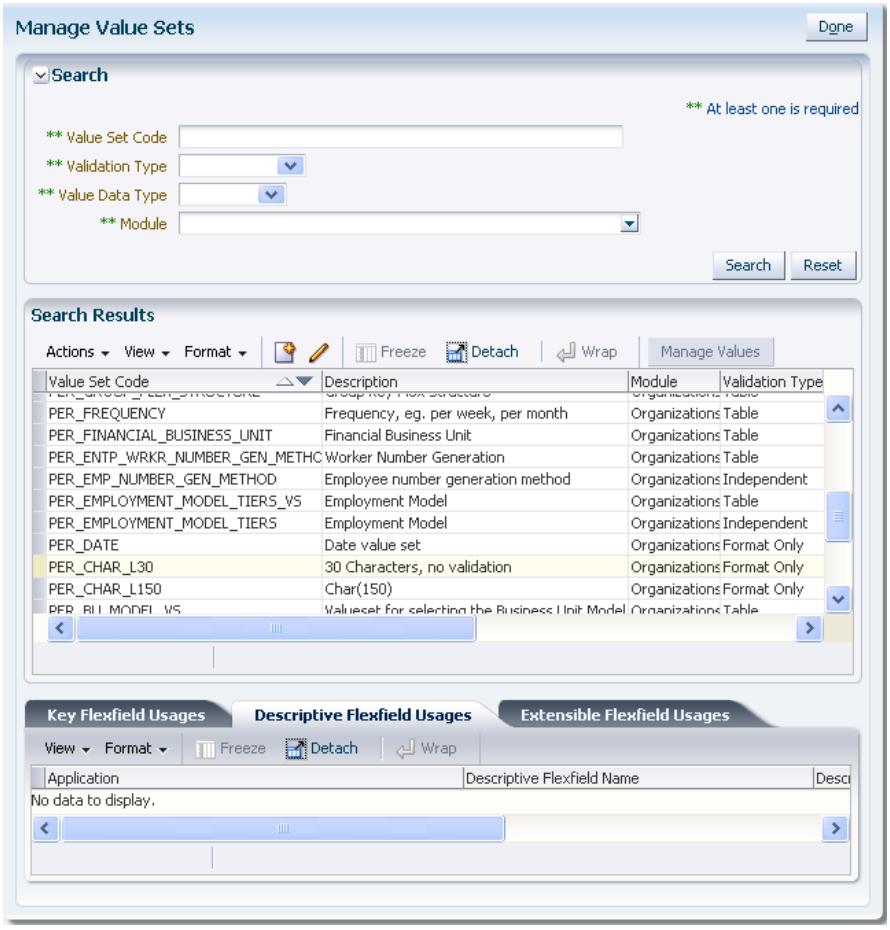


Figure 5–5 Manage Extensible Flexfields Task



You must specify validation rules for your custom attributes, such as minimum and maximum values or a list of valid values. Use the Manage Value Sets task to specify validation rules, as shown in Figure 5–6.

Figure 5–6 Manage Value Sets Task



Perform the following steps to configure custom attributes. These steps are described in detail in the remaining sections of this chapter.

1. Find the flexfields on the page.
2. Plan the flexfield configuration.
3. Find or create the required value sets using the Manage Values Sets task and optionally configure value set security privileges.
4. Define the attributes using the appropriate task — Manage Extensible Flexfields or Manage Descriptive Flexfields.
5. Optionally, validate the flexfield configuration.
6. Deploy the flexfield to display the custom attributes on the application pages and to make them available for integration into the Oracle Fusion Middleware technology stack. The flexfield artifacts that are generated during deployment are saved to an Oracle Metadata Services (MDS) repository. You can optionally deploy the flexfield to a sandbox for initial testing.
7. Perform the necessary steps to integrate the custom attributes into the technology stack.

After your custom attributes are displayed on the application pages, you can customize the attributes on a per-page basis using Page Composer, as described in Chapter 3, "Customizing Existing Pages." For example, you can hide a custom attribute, you can change its prompt or other properties, and you can reorder the

custom global attributes so that they are interspersed with the core attributes in the same parent layout.

If you need to create translations of the value sets and the custom attributes for different locales, see Chapter 16, "Translating Custom Text."

5.1.2 What You Cannot Do with Flexfields at Runtime

You can use the flexfield feature to add attributes only to business objects that have a descriptive or extensible flexfield. For information about creating flexfields for other business objects, see the "Getting Started with Flexfields" chapter in the *Oracle Fusion Applications Developer's Guide*.

Note: You cannot use key flexfields to add custom attributes.

The following applications, which are part of the Customer Relationship Management (CRM) product family, include CRM Application Composer for extending and customizing applications. With those applications, use CRM Application Composer to add custom attributes. For all other applications, use flexfields to add custom attributes, as described in this chapter.

- Sales
- Marketing
- Customer Center
- Trading Community Architecture
- Order Capture

For more information, see Section 4.2, "Editing Objects."

5.1.3 Before You Begin Using Flexfields to Create Custom Attributes

Before you use flexfields to create custom attributes, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

5.2 Finding the Flexfields on a Page

Before you begin the planning stage, you must first determine what type of flexfields — extensible or descriptive — are available for customizing. You also need the flexfield codes. If you are working with an extensible flexfield, you must know the flexfield's usage. Use Page Composer to find the flexfields on a page. For descriptive flexfields, you can also choose **Highlight Flexfields** from the Administration menu to highlight the flexfields on a page.

Task: Find Descriptive Flexfields on a Page

To obtain information about the descriptive flexfields on a page, open the page and select **Highlight Flexfields** from the **Administration** menu. Hover over the information icon next to the highlighted fields to display information about the flexfield, such as the flexfield code. Select **Unhighlight Flexfields** from the Administration menu when you no longer want to see the highlighted flexfields.

You can also use Page Composer to get information about the descriptive flexfields on a page. Open the page in Source view and look for `<descriptiveFlexfield>` elements. Open the properties panel for the element to view the flexfield name and flexfield code.

Note: Not all flexfields are available for creating custom attributes. Consult the product-specific documentation in Oracle Fusion Applications Help to verify whether there are any restrictions on using the flexfield.

Task: Find Extensible Flexfields on a Page

Use Page Composer to get information about extensible flexfields on a page. Open the page in Source view and look for a region that is bound to an EffContextsPageContainer task flow. Open the properties panel for the region to view the flexfield name, flexfield code, and usage.

Note: Not all flexfields are available for creating custom attributes. Consult the product-specific documentation in Oracle Fusion Applications Help to verify whether there are any restrictions on using the flexfield.

5.3 Planning Your Flexfields

The planning of your custom attributes is an important activity in preparation for flexfield configuration. The way in which you plan and configure a flexfield depends on whether it is descriptive or extensible.

5.3.1 Planning Descriptive Flexfields

If you are configuring a descriptive flexfield, complete the following tasks:

1. Identify existing context values
2. Identify whether the context value is derived
3. Identify available parameters
4. List custom attributes
5. Plan the descriptive flexfield structure
6. Define attribute properties
7. Map attributes to available table columns
8. Define validation rules

Task: Identify Existing Descriptive Flexfield Context Values

To identify existing context values, access the Manage Descriptive Flexfields task by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task. Next, search for and edit the flexfield to view the list of configured context values.

If context values have been preconfigured, you should consult the product-specific documentation in Oracle Fusion Applications Help for information about the use of those values.

Task: Evaluate Whether the Context Value is Derived

The context value for a descriptive flexfield might have been preconfigured to be derived from an external reference. For example, if the context is Marriage Status, the value might be derived from an attribute in the employee business object. When the context value is derived, you might need to take the derived values and their source into consideration in your plan. To determine whether the context value is derived, look at the **Derivation Value** field in the Context Segment region in the Edit Descriptive Flexfields window for the flexfield. For more information about derived values, see Task: Identify Available Descriptive Flexfield Parameters.

Task: Identify Available Descriptive Flexfield Parameters

Some descriptive flexfields provide parameters, which enable you to set the initial value of an attribute from external reference data, such as a column value or a session variable. For example if a flexfield has a user email parameter, you would be able to configure the initial value for a customer email attribute to be derived from the parameter. You can view the **Derivation Value** dropdown list in the Create Segment window for the flexfield to see what parameters are available for a descriptive flexfield. If you decide to use one of the parameters to set an initial value, you would select that parameter from the dropdown list when you add the attribute during the flexfield configuration process described in Section 5.5.1, "Configuring Descriptive Flexfields."

Task: List Custom Attributes

List all the custom attributes that you want to add. Later, as described in Task: Define Attribute Properties, you define the attribute details.

Task: Plan the Descriptive Flexfield Structure

When you configure the flexfield, you will use the flexfield's *global*, *context*, and *context-sensitive segments* to structure your custom attributes. For the custom attributes that you want to store for every instance of the business object, such as a custom attribute that you want to store for every job, you will add them as global segments when you configure the flexfield, as described in Section 5.5.1, "Configuring Descriptive Flexfields."

For the custom attributes that you want to store based on a context (a condition or situation), define the context and list the valid values for the context. For example, you might want to use the Job Attributes descriptive flexfield on the Manage Jobs page to display different custom attributes depending on whether the job is for a service technician, an engineer, or a driver. You would define the context as Category, and you would list Service Technician, Engineer, and Driver as the list of valid values for the Category context. Next, list the custom attributes that you want to store for each of the valid values. For example, you might store the service type custom attribute for a service type job and you might store the commercial driver's license (CDL) class for a driver job. Later, when you configure the flexfield, you would set the prompt for the context segment to Category, you would create the Service Technician, Engineer, and Driver *context values*, and you would define the context-sensitive segments for each of the three context values. For example, you would create a service type context-sensitive segment for the Service Technician context value and you would create a commercial driver's license (CDL) class context-sensitive segment for the Driver context value.

There is only one context segment available for descriptive flexfields. If you have more than one group of custom attributes where you could use the context segment, you will have to pick one group over the others, based on your company's needs and priorities, and add the other custom attributes as global segments.

The following list shows an example of a structural plan for the Job Attributes descriptive flexfield:

- Global segments
  - offsite: YES or NO
- Context segment
  - Prompt: Category
  - Value set: JOB_CATEGORIES
  - Context values:
    * Service Technician
    * Engineer
    * Driver
- Context-sensitive segments for Service Technician
  - service type: such as appliance, heating, ventilation, and air conditioning (HVAC), facilities maintenance, and vehicle maintenance
- Context-sensitive segments for Engineer
  - regulations: such as building codes and Occupational Safety and Health Administration (OSHA) regulations
- Context-sensitive segments for Driver
  - commercial driver's license (CDL) class: such as A and B

Task: Define Attribute Properties

For each custom attribute that you want to add, define the attribute properties that are listed in Table 5–1.

Table 5–1 Attribute Properties

Property	Description
Sequence	The order the attribute appears in relation to the other configured attributes.
Enabled	Whether the attribute can be used.
Data type	Character, date, date and time, or number.
Range type	If the attribute is part of a range specification, whether the attribute is the low value or the high value. For example, if adding minimum height and maximum height attributes, the minimum height attribute has a range type of low.
Required	Whether the end user is required to provide a value.
Initial value	The initial default value for an attribute when the row is created. You can specify a constant value or, for descriptive flexfields, you can specify a flexfield parameter, which provides a derived value.
Derivation value	The flexfield parameter from which to derive the attribute's value. Whenever the parameter value changes, the attribute's value is changed to match. If you derive an attribute value from a descriptive flexfield parameter, consider making the attribute read-only, because user-entered values are lost whenever the parameter value changes.

Table 5–1 (Cont.) Attribute Properties

Property	Description
Prompt	The string to be used for the attribute's label in the user interface. Note that for global and context-sensitive attributes, you store the prompt string in the **Short Prompt** field in the manage flexfield task. The **Prompt** field is not utilized in the user interface and is reserved for future use.
Display type	The type of field in which to display the attribute, such as text box, dropdown list, chooser (checkbox), list of values, radio button group, or hidden.
Checked and unchecked values	If the display type is chooser (checkbox), the actual values to save. For example, Y and N or 0 and 1.
Display size	The character width of the field.
Display height	The height of the field.
Read only	Whether the field should display as read-only (non-editable) text.

Task: Map Attributes to Available Descriptive Flexfield Table Columns

Compile a list of available flexfield table columns and choose which column to use for each custom attribute.

> **Tip:** For descriptive flexfields, first map your global segments. Then you can allocate the remaining table columns to the context-sensitive segments. You can allocate a table column to more than one context-sensitive segment as long as each context-sensitive segment is for a different context value.

To see the available table columns for a data type, edit the flexfield in the Manage Descriptive Flexfields task and access the Create Segment window. Select the desired data type, view the **Table Column** list as shown in Figure 5–7, and note the available columns for that data type. Repeat for each data type for which you will be adding attributes and map the custom attributes to available columns.

Figure 5–7 Table Column List for the Character Data Type

Task: Define Validation Rules For Descriptive Flexfield Custom Attributes

Define the validation rules for the custom attributes. For example one attribute might need to match a specified format, while another attribute might be restricted to a list of values. You use value sets to specify the validation rules for an attribute, as described in Section 5.4, "Creating Custom Value Sets." Define each attribute's validation rules and check if value sets exist for those rules or you must create new ones.

Note that validation for a descriptive flexfield context attribute is somewhat different from validation of global and context-sensitive attributes. While value sets are required for global and context-sensitive attributes, a value set is optional for descriptive flexfield context attributes. If a value set is not defined for a context

attribute, the application validates an input value against the context attribute's values. For example, if you created YES and NO context-attribute values, you do not need an associated value set if YES and NO are the only valid values. However, if the list of valid values for a context is a superset of the context attribute's values, then a value set is required. For example, suppose, for example, that you need custom context-sensitive attributes for a YES value, but you do not need any custom attributes for a NO value. You need to add only a YES context attribute value. Because you do not have a NO context value, the context attribute requires a value set of YES and NO, as both values are valid. A value set is also required when the valid values are a subset of the context values. For example, you might have contexts for several countries, but the list of values from which the end user selects the context value might be confined to the countries in the user's region.

You can use only *table* and *independent* value sets to validate context values. If you use a table value set, you cannot reference other flexfield segments in the value set's WHERE clause. That is, the WHERE clause cannot reference SEGMENT.*segment_code* or VALUESET.*value_set_code*. For information about table and independent value sets, Section 5.4, "Creating Custom Value Sets."

When determining an attribute's validation rules, consider the following questions:

- What is the data type — character, date, date and time, or number?
- Does the attribute require any validation beyond data type and maximum length?
- If the data type is character, should the value be restricted to digits, or are alphabetic characters allowed? If a number, should the value be zero filled?
- If alphabetic characters are allowed, should they automatically be changed to uppercase?
- For numeric values, how many digits can follow the radix separator?
- Do you want to provide a list of values from which to select a valid value for the attribute? If so, consider the following questions:
  - Can you use an existing application table from which to obtain the list of valid values, or do you need to create a custom list?
  - If you are using an existing table, do you need to limit the list of values using a WHERE clause?
  - Does the list of valid values depend on the value in another flexfield attribute?
  - Is the list of valid values a subset of another flexfield attribute's list of values?
- Does the value need to fall within a range?

5.3.2 Planning Extensible Flexfields

If you are configuring an extensible flexfield, complete the following tasks:

1. Identify the extensible flexfield's category hierarchy structure
2. Identify existing context values
3. List custom attributes
4. Plan categories
5. Plan the extensible flexfield structure
6. Define attribute properties
7. Map attributes to available table columns

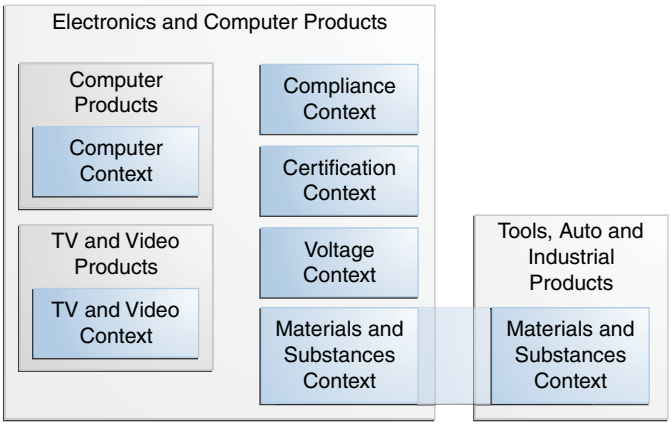
8. Define validation rules

Task: Identify the Extensible Flexfield's Category Hierarchy Structure

Most extensible flexfields are shipped with only one *category*, which is often referred to as the *root* category, but there are a few extensible flexfields that are preconfigured with category hierarchies. In addition, some Oracle Fusion applications provide user interfaces to create and manage an extensible flexfield's category hierarchy, as described in the product-specific documentation in the Oracle Fusion Applications Help. If a category hierarchy exists for the flexfield, you can take advantage of the hierarchy to reuse contexts for similar entities, such as similar items in a product catalog.

The Item Extended Attributes flexfield shown in Figure 5–9 is an example of a flexfield that uses the category hierarchy feature to reuse contexts for all electronic and computer items, such as televisions, video equipment, and computers. The flexfield's Electronics and Computers category has been customized to contain contexts for compliance and certification, voltage, and materials and substances, as shown in Figure 5–8. The TV and Video subcategory and the Computers subcategory *inherit* the Electronics and Computer contexts in addition to having their own contexts.

Figure 5–8 Category Hierarchy

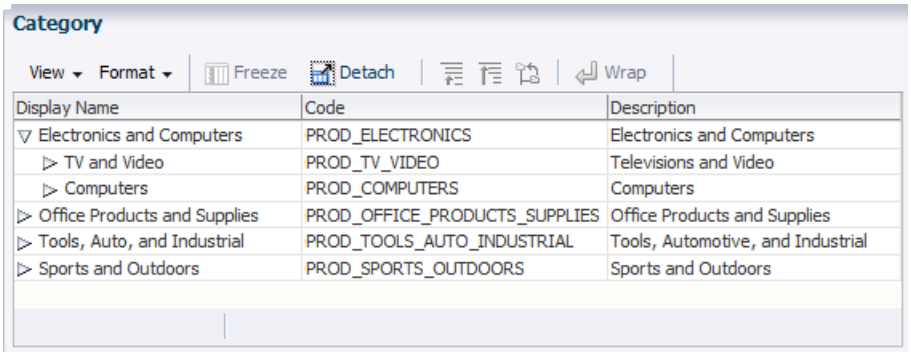


Contexts are reusable within a given extensible flexfield. For example, in Figure 5–8 the Materials and Substances context belongs to both the Electronics and Computer Products category and the Tools, Auto, and Industrial Products category.

If you were adding custom attributes to the Item Extended Attributes flexfield, you would plan how to work them into the existing category hierarchy.

You can use the category hierarchy feature in your flexfield plan, as described in Task: Plan Extensible Flexfield Categories, only if a category hierarchy exists for the flexfield. To view whether a category hierarchy exists for an extensible flexfield, access the Manage Extensible Flexfields task by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task. Next, search for and edit the flexfield to see its categories. For example, Figure 5–9 shows the Category region for the Item Extended Attributes flexfield.

Figure 5–9 Category Region



Task: Identify Existing Context Values

Some Flexfields have preconfigured context values. To identify a flexfield's existing context values, display the user interface page or pages that contain the flexfield segments and look for region headers, which identify existing contexts. For example, the first region in Figure 5–3 contains the attributes that belong to the Recovery and Recycling context value.

You can also use the Manage Extensible Flexfields task to view the list of configured context values.

If context values have been preconfigured, then consult the product-specific documentation in Oracle Fusion Applications Help for information about the use of these contexts.

Task: List Custom Attributes

List all the custom attributes that you want to add using the extensible flexfield. Later, as described in Task: Define Attribute Properties, you define the attribute details.

Task: Plan Extensible Flexfield Categories

All extensible flexfields have at least one category, but some flexfields have been set up with a hierarchy of categories, as described in Task: Identify the Extensible Flexfield's Category Hierarchy Structure.

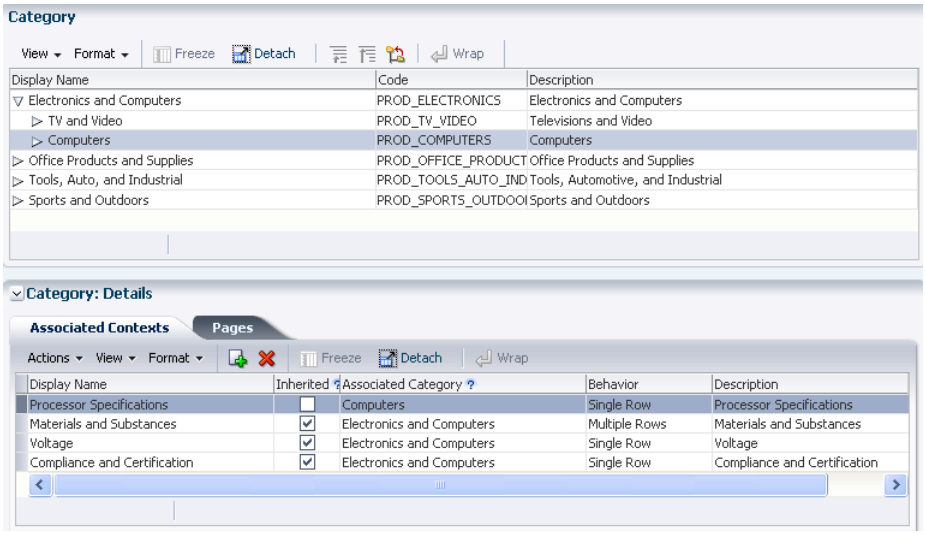
How you structure the flexfield configuration depends upon the way in which categories are defined for the flexfield. Most extensible flexfields are preconfigured with one category, and you associate all your contexts and pages with that category. Other extensible flexfields are preconfigured with several categories and you associate your contexts and pages with those categories as instructed by the product-specific documentation in Oracle Fusion Applications Help. A small number of extensible flexfields enable you to configure multiple categories by using provided user interfaces. In these cases, you can take advantage of the inheritance feature described in Task: Identify the Extensible Flexfield's Category Hierarchy Structure to associate a context with more than one category. For example, the Item Extended Attributes flexfield might be set up with the following category hierarchy:

- Electronics and Computers
  - TV and Video
  - Computers

Suppose you want to store voltage information for all electronic and computer items. If you associate a Voltage context with the Electronics and Computers category, then both the TV and Video subcategory and the Computers subcategory inherit the

Voltage context from the parent Electronics and Computers category. Figure 5–10 shows the contexts that have been associated with a Computers subcategory. Three of the contexts are inherited from the parent category.

Figure 5–10 Contexts Associated with the Computer Category



Task: Plan the Extensible Flexfield Structure

Extensible flexfields enable you to group similar custom attributes into *contexts*, as illustrated in Figure 5–2. A context's attributes are displayed together in a region, and the region's header is the context value. For example, the Item Extended Attributes flexfield might have the following contexts:

- Materials and Substances
- Compliance and Certification
- Voltage

On the other hand, for the Position EIT Information flexfield, you might group your custom attributes into the following contexts:

- Educational Requirements
- Certification and License Requirements
- Travel Requirements

To begin the planning process, group the custom attributes into contexts and determine the order in which the attributes should appear.

A context can optionally store multiple rows of data for a single entity, such as a specific job or position. For example, with the Certification and License Requirements context for the Position EIT Information flexfield, you might want to store values for all the certificates and licenses that are required to perform each position. For contexts that store multiple rows, decide how to uniquely identify each row. That is, identify which attributes form a unique key. For example, you might decide that the combination of certificate type and certificate name uniquely identifies a given row for the Certification and License Requirements context.

For each category (or for the single category if the flexfield was not set up with multiple categories), next group the category's contexts into logical pages and determine the sequence in which the logical pages should appear.

Note: For hierarchical categories, the child categories inherit the logical pages that are defined for the parent categories.

The following list shows an example plan for custom computer attributes for the Item Extended Attributes flexfield. In this example, the Electronics Information page is inherited from the parent Electronics and Computers category.

- Page: Electronics Information
  - Context: Compliance and Certification, single row
    - ISO 14001 (International Organization for Standardization for an Environmental Management System)
    - ENERGY STAR (energy efficiency guidelines)
    - ROHS (Restriction of the use of certain hazardous substances in electrical and electronic equipment)
  - Context: Voltage, single row
    - minimum voltage
    - maximum voltage
    - current type
  - Context: Materials and Substances, multiple rows
    - material
    - contain recyclate
    - percent unit mass
- Page: Computer Information
  - Context: Processor Specifications, single row
    - manufacturer
    - CPU type
    - processor interface
    - processor class
    - processor speed
    - cores

The following list shows a sample plan for the Position EIT Information flexfield:

- Page: Additional Position Information
  - Context: Educational Requirements, single row
    - level (high school, bachelor, master, MD, Ph.D.)
  - Context: Certification and License Requirements, multiple rows
    - type (certificate or license)
    - name (for example, Automotive Service Excellence, NACE International Level II Coating Inspector, Cathodic Protection Specialist)
  - Context: Travel Requirements, single row
    - overnight travel required

* international travel required

Task: Define Attribute Properties

For each custom attribute that you want to add, define the attribute properties that are listed in Table 5–1.

Also define the indexed property for each attribute. This indicates whether the attribute should be marked as selectively required in search panels. That is, whether it is one of the attributes for which an end user must enter a value before conducting a search. Note that if you mark an attribute as indexed, you must ask your database administrator to create an index on the segment column that you configure for that custom attribute as described in Section 5.5.2, "Configuring Extensible Flexfields.".

Task: Map Attributes to Available Extensible Flexfield Table Columns

Compile a list of available flexfield table columns and, for each context, choose which column to use for each custom attribute.

> **Tip:** For extensible flexfields, if you do not have enough available segments of a certain type (character or number) for a context, add another context to the same category to hold the remaining attributes.

To see the available table columns for a data type, edit the flexfield in the Manage Extensible Flexfields task and access the Create Segment window. Select the desired data type, view the **Table Column** list as shown in Figure 5–7, and note the available columns for that data type. Repeat for each data type for which you will be adding attributes for a given context and map the context's custom attributes to the available columns.

Task: Define Validation Rules for Extensible Flexfield Custom Attributes

Define the validation rules for every attribute as described in Task: Define Validation Rules For Descriptive Flexfield Custom Attributes.

5.4 Creating Custom Value Sets

You use the following types of *value sets* to control the values that can be stored for the custom attributes:

- Table
- Independent
- Dependent
- Subset
- Format only

A value set is a predefined group of values that you can use to validate custom attributes. Different custom attributes in the same flexfield can use the same value set, and custom attributes in different flexfields can share value sets.

Value sets enable you to enforce the following types of data validation:

- List of values: You can use one of the following types of lists to specify the valid values for an attribute:
  - Table column: If the valid values exist in a table column, use a *table* value set to specify the list of values. To limit the valid values to a subset of the values in the table, use a SQL WHERE clause. Table value sets also provide some

advanced features, such as enabling validation to depend upon custom attributes in the same structure.

- Custom list: Use an *independent* value set to specify a custom set of valid values. For example, you can use an independent value set of Mon, Tue, Wed, and so forth to validate the day of the week.
- Dependent custom list: Use a *dependent* value set when the available values in the list and the meaning of a given value depend on which independent value was selected for a prior custom attribute. For example the valid holidays depend on which country you are in. A dependent value set is a collection of value subsets, with one subset for each value in a corresponding independent value set.

You can further limit the valid values that an end user can select or enter by specifying format, minimum value, and maximum value.

- Subset: Use a *subset* value set when you want to use a subset of values from an existing independent value set. For example, if you have an independent value set for the days of the week, a weekend subset can be composed of its entries for Saturday and Sunday.
- Format: Use a *format-only* value set when you want to allow end users to enter any value so long as that value conforms to formatting rules. For example, if you specify a maximum length of 3 and numeric-only, then end users can enter 456, but not 4567 nor 45A. You can also specify the minimum and maximum values, whether to right-justify, and whether to zero-fill. With a format-only value set, no other types of validation are applied.
- Range of values: You can use either a format-only, independent, or dependent value set to specify a range of values. For example, you might create a format-only value set with format type of Number where the end user can enter only the values between 0 and 100. Or, you might create a format-only value set with format type of Date where the end user can enter only dates for a specific year (a range of 01–JAN–93 to 31–DEC–93, for example). Because the minimum and maximum values enforce these limits, you need not define a value set that contains each of these individual numbers or dates.

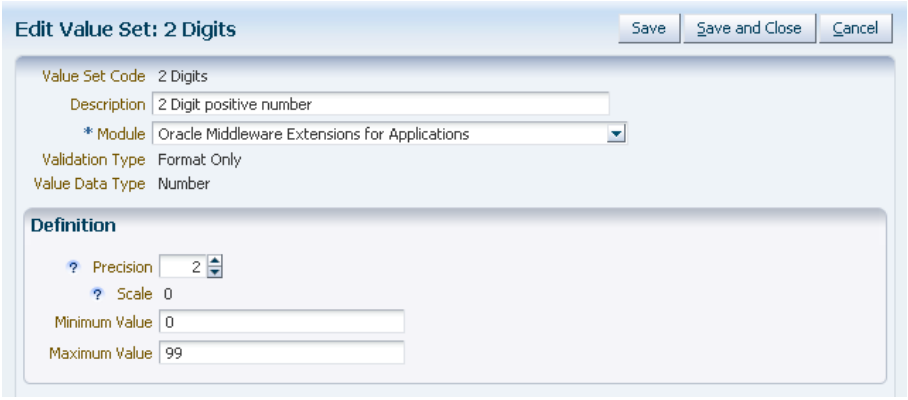
Note: You can use only table and independent value sets to validate context values. The data type must be character and the maximum length of the values being stored must not be larger than the context's column length. If you use a table value set, the value set cannot reference flexfield segments in the value set's WHERE clause other than the flexfield segment to which the value set is assigned. You learn about assigning value sets to flexfield segments in Section 5.5, "Configuring Flexfields."

If you are creating an independent, dependent, or subset value set, you must also define the set of valid values. For table, independent, dependent, and subset value sets, you can optionally implement value set security. If the Oracle Fusion applications are running in different locales, you might need to provide different translations for the values and descriptions. For more information, see Section 16.5, "Translating Flexfield and Value Set Configurations."

Use the Manage Value Sets task as shown in Figure 5–11 to create and manage value sets. For more information, see the "Manage Value Sets" section in the *Oracle Fusion Applications Common Implementation Guide*. In Section 5.5, "Configuring Flexfields" you

learn how to assign the value sets to the flexfield segments that you configure for your custom attributes.

Figure 5–11 Edit Value Set Page in the Manage Value Sets Task



> **Note:** The management of value sets cannot be performed in a sandbox. For a list of the customizations that can be performed in a sandbox, see Chapter 2.2, "Using the Sandbox Manager."

When you change an existing value set, the deployment status for all affected flexfields changes to Edited. You must redeploy all flexfields that use that value set to make the flexfields reflect the changes. The **Usages** tabs show which flexfields are affected by the value set changes. You learn about deploying flexfields in Section 5.7, "Deploying Flexfield Configurations."

Before you begin:

You will need to do the following before you can begin creating custom value sets:

- Ensure that you have the necessary function and data security privileges to access the Manage Value Sets task. Contact your security administrator for details.
- Access the Manage Value Sets task by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task.

Task: Define Format Specifications

Regardless of which type of validation you use for an attribute, the attribute will most likely require some sort of format specification. Before you create a value set, consider how you will specify the required format. Depending on the validation type and the value data type, you will be able to specify one or more of the options shown in Table 5–2.

Table 5–2 Format Options

Option	Description
Value data type	Character, Number, Date, Date Time.

Table 5–2 (Cont.) Format Options

Option	Description
Value subtype	Text, Translated text, Numeric digits only, Time (20:08), Time (20:08:08). An additional data type specification for the Character data type for the Dependent, Independent, and Format validation types. See the Caution note following this table for information about the Text and Translated text subtypes.
Maximum length	Maximum number of characters or digits for Character data type.
Precision	Maximum number of digits the user can enter.
Scale	Maximum number of digits that can follow the decimal point.
Uppercase only	Lowercase characters automatically changed to uppercase.
Zero fill	Automatic right-justification and zero-filling of entered numbers (affects values that include only the digits 0-9).

Caution: When choosing between a value subtype of Text or Translated text, choose Translated text if your application has more than one language installed, or there is any possibility that you might install additional languages, and you might want to translate the display values into the other languages. Choosing the Translated text subtype does not require you to provide translated values now, but you cannot change this option if you decide to provide them later.

Task: Create a Format-Only Value Set

If you do not need to validate input against a list of valid values, then create a value set with the **Validation Type** of Format. You can also use this validation type to create a range specification, such as all numbers between 1 and 100.

Task: Create an Independent Value Set

If you need to validate the input against a custom list of values, where the list is not stored in an application table and the values are not dependent upon or a subset of another independent value set, then create a value set with a **Validation Type** of Independent.

After you create the value set, define the set of valid values as described in Task: Define the Set of Valid Values.

Note: If the independent value set has dependent value sets, then define the dependent values sets before you define the valid values for the independent value set.

Task: Create a Dependent or Subset Value Set

If you need to validate the input against a custom list of values where the values are dependent upon or members of an independent value set, then create a value set with a **Validation Type** of Dependent or Subset, as appropriate. You must specify the independent value set on which it depends.

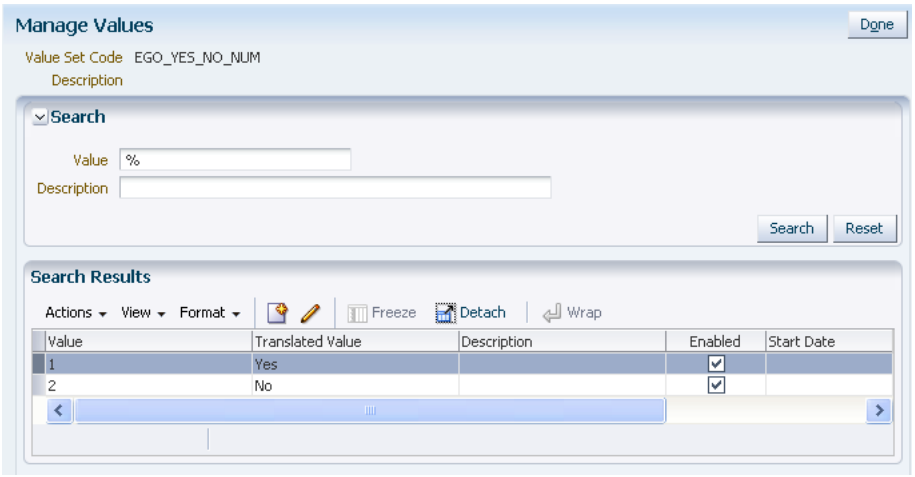
After you create the value set, define the set of valid values as described in Task: Define the Set of Valid Values.

Note: If the independent value set has dependent value sets, then define the dependent values sets before you define the valid values for the independent value set.

Task: Define the Set of Valid Values

Independent, dependent, and subset value sets require a customized list of valid values. Use the Manage Values page as shown in Figure 5–12 to create and manage the valid values for a value set. To access the Manage Values page, select the value set from the Manage Value Sets page and click **Manage Values**.

Figure 5–12 Manage Values Page



If you are adding a value to a dependent value set, you must associate it with a value from the parent independent value set.

If you are adding a value to a subset value set, you must select it from the parent independent set.

If the value set's subtype is Translated text, then you can enter a translated value for a locale by logging in using that locale (or choosing **Set Preferences** from the **Personalization** menu in the global area) and entering the translated value. If you do not provide a translation for a given locale, then the value that was first entered is used for that locale.

Task: Create a Table Value Set

If you need to validate the input against a list of values from an application table, then create a value set with the Table validation type. You define which table you want to use and you specify the column that contains the valid value. You can optionally specify the description and ID columns, a WHERE clause to limit the values to use for your set, and an ORDER BY clause.

If you specify an ID column, the flexfield saves the ID value, instead of the value from the value column, in the associated flexfield segment. You learn about assigning values sets to flexfield segments in Section 5.5, "Configuring Flexfields."

If the underlying table supports translations, you can enable the display of translated text by basing the value set's value column on a translated attribute of the underlying table. You should also define an ID column that is based on an attribute that is not language-dependent so that the value's invariant ID is saved in the transaction table.

This allows the runtime to display the corresponding translated text from the value column for the runtime session's locale. For more information, see "Using Multi-Language Support Features" in the *Oracle Fusion Applications Developer's Guide*.

After you assign a value set to a flexfield, you can use the following bind variables in the WHERE clause.

- **:{SEGMENT.***segment_code***}**

  This bind variable refers to the ID (if the value set is ID-validated) or value (if not ID-validated) of a segment where *segment_code* identifies the segment. The data type of the bind value is the same as the data type of the segment's column.

  The segment must have a sequence number that is less than the sequence number of the segment with this bind variable. A matching segment must exist in the current flexfield context.

  This bind variable is useful when the set of valid values depends on the value in another segment. For example, the values to select from a CITIES table might depend upon the selected country. If SEGMENT1 contains the country value, then the WHERE clause for the CITIES table might be `country_code = :{SEGMENT.SEGMENT1}`.

- **:{VALUESET.***value_set_code***}**

  This bind variable refers to the ID (if the value set is ID-validated) or value (if not ID-validated) of the segment that is assigned to the value set that is identified by the *value_set_code*. The data type of the bind value is the same as the data type of the segment's column.

  The segment must have a sequence number that is less than the sequence number of the segment with this bind variable. If more than one segment is assigned to the value set, the closest prior matching segment will be used to resolve the bind expression. A matching segment must exist in the current flexfield context.

  This bind variable is useful when the set of valid values depends on the value in another segment and that segment code can vary, such as when the value set is used for more than one context or flexfield. For example, the values to select from a CITIES table might depend upon the selected country. If the value set for the segment that contains the country value is COUNTRIES, then the WHERE clause for the CITIES table might be `state_code = :{VALUESET.COUNTRIES}`.

- **:{FLEXFIELD.***internal_code***}**

  This bind variable refers to an internal code of the flexfield in which the value set is used, or to a validation date. The *internal_code* must be one of the following:

  - `APPLICATION_ID` — the application ID of the flexfield in which this value set is used. The data type of `APPLICATION_ID` and its resulting bind value is NUMBER.
  - `DESCRIPTIVE_FLEXFIELD_CODE` — the identifying code of the flexfield in which this value set is used. The data type of `DESCRIPTIVE_FLEXFIELD_CODE` and its resulting bind value is VARCHAR2. Note that you use this string for both descriptive and extensible flexfields.
  - `CONTEXT_CODE` — the context code of the flexfield context in which this value set is used. The data type of `CONTEXT_CODE` and its resulting bind value is VARCHAR2.

  - SEGMENT_CODE — the identifying code of the flexfield segment in which this value set is used. The data type of SEGMENT_CODE and its resulting bind value is VARCHAR2.
  - VALIDATION_DATE — the current database date. The data type of VALIDATION_DATE and its resulting bind value is DATE.

- **:{PARAMETER.***parameter_code***}**

  This bind variable refers to the value of a flexfield parameter where *parameter_code* identifies the parameter. The data type of the resulting bind value is the same as the parameter's data type.

> **Note:** You cannot assign a table value set to a context segment if the WHERE clause uses VALUESET.*value_set_code* or SEGMENT.*segment_code* bind variables.

Task: Implement Value Set Security

If you need to control end user access to the values in an independent, dependent, subset, or table value set, then you can create security *policies* for that value set. These policies enable you to specify what values each end user can enter and view, based on the users' *roles*. End users will not be able to enter values for which they do not have access, and only the values they have access to appear in the list of values for the associated segments. Any security rules that you define for a value set affect every segment that uses the value set.

A *resource* identifies the data that must be secured by the policies, which in this case is the value set. Value set data security follows a "deny all, allow some" approach. This means that access to all of a resource's data is denied by default and, based on policies, end users have access only to the data that is appropriate for their role.Use *conditions* to define the data that can be accessed and use *policies* to identify the roles that have the authority to access the data that is identified by the condition.

You can implement value set security after you create the value set, or you can implement it later by editing the value set. To implement security, select the **Security Enabled** checkbox and then provide the data security resource name. You can enter the name of an existing resource or type the name of a data security resource that you want to create. This will be the name used for the database resource in the data security system. The name typically matches the code value for the value set.

> **Note:** You cannot edit the data security resource name after you save your changes.

After you save your changes to the Data Security Resource Name, you can optionally click the **Edit Data Security** button to access the Edit Data Security page, provided that you have access to the Manage Database Security Policies task. On the Edit Data Security page, you can specify conditions, such as filters or SQL predicates, and you can define policies where you associate roles with conditions. You can use a filter for simple conditions. For more complex conditions, use a SQL predicate.

The data security conditions and policies that you define are similar to those that you can define for business objects. Value set security policies differ in the following ways:

- You can grant only read access to end users.You cannot specify any other action.
- When defining a condition that is based on a SQL predicate, use VALUE, VALUE_NUMBER, VALUE_DATE, VALUE_TIMESTAMP, or VALUE_ID to reference the value

from a dependent, independent, or subset value set. For table value sets, use a table alias to define the table, such as `&TABLE_ALIAS.category_id=70`.

For more information data security resources, roles, conditions, and policies, see the "Managing Oracle Fusion Applications Data Security Policies" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

5.5 Configuring Flexfields

Use the Manage Descriptive Flexfields task or the Manage Extensible Flexfields task, depending on the flexfield type, to configure a flexfield.

5.5.1 Configuring Descriptive Flexfields

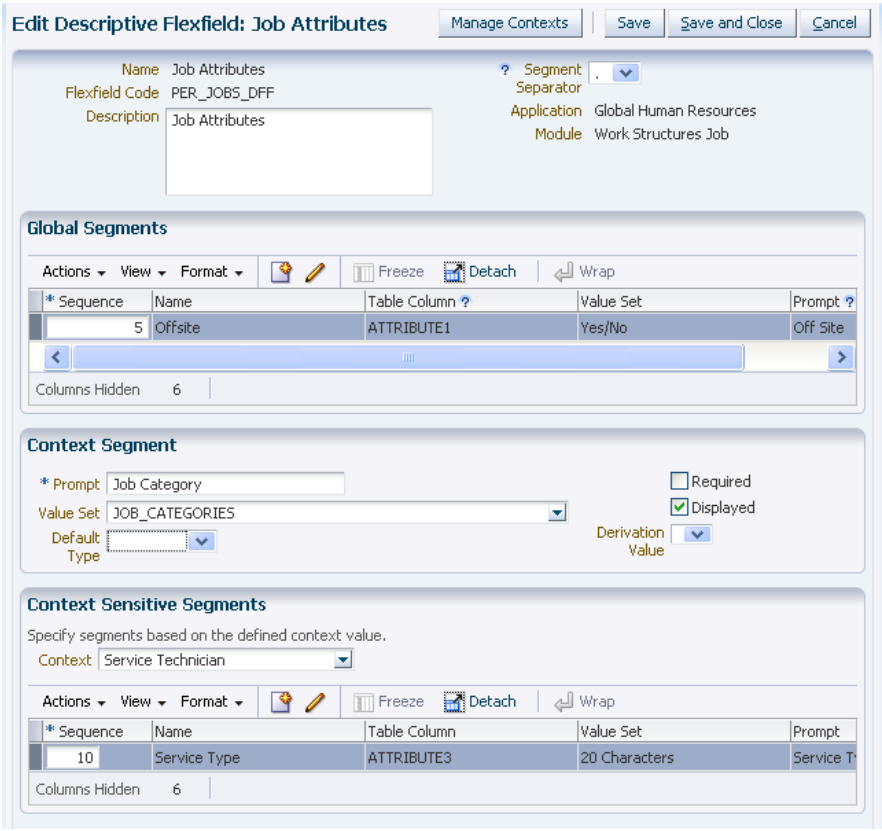
Use the Manage Descriptive Flexfields task shown in Figure 5–13 to configure a descriptive flexfield.

When you configure a descriptive flexfield, you define the global segments, the context segment, and the context-sensitive segments. For each segment, you define its display properties and you specify how to validate its values.

Note: A descriptive flexfield might be used for more than one application table. For example, a flexfield might be associated with a USER table and a USER_HISTORY table. These associations are called flexfield *usages*. When you configure a flexfield, the configuration applies to all its usages.

If the Oracle Fusion applications are running in different locales, then you might want to provide different translations for the translatable text, such as prompts and descriptions. To create the translations, log in with each of the different locales (or choose **Set Preferences** from the **Personalization** menu in the global area to set the locale) and change the text to the translated text for that locale. For more information, see Section 16.5, "Translating Flexfield and Value Set Configurations."

Figure 5–13 Edit Descriptive Flexfield Page in the Manage Descriptive Flexfields Task



For more information, see the "Manage Descriptive Flexfields" section in the *Oracle Fusion Applications Common Implementation Guide*.

Before you begin:

You will need to do the following before you can begin configuring the descriptive flexfield:

- Plan your flexfield configuration as described in Section 5.3, "Planning Your Flexfields."
- Create the required value sets as described in Section 5.4, "Creating Custom Value Sets."
- Compile a list of the UI pages that are affected by the descriptive flexfield as well as other artifacts in the Oracle Fusion Middleware technology stack. See Section 5.8, "Integrating Custom Attributes" for more information about how flexfield configuration affects these artifacts. Using this list, plan how you will integrate and test your flexfield configuration
- Ensure that you have the necessary function and data security privileges to access the Manage Descriptive Flexfields task. Contact your security administrator for details.
- Access the Manage Descriptive Flexfields task by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task.

Task: Configure Global Segments

Using the information that you gathered in your planning stage, configure a global segment for every global attribute that you identified in Task: Plan the Descriptive Flexfield Structure in Section 5.3.1, "Planning Descriptive Flexfields." For each segment, provide the identifying information, the column assignment, how the value should be validated, the initial default value, and the display properties.

Note: The segment code is used in the flexfield's element in the XML schema for web services. The use of leading numeric characters, spaces, underscores, and multibyte characters, which are all encoded in XML schemas, make the code in the schema element difficult to read.

Tip: When you create segments for your attributes, you specify the sequence number for the segment. This sequence affects the order in which the attribute is displayed on the page. Consider numbering the segments in multiples, such as 4, 5, or 10, to make it easy to insert new attributes.

Task: Configure Contexts

On the Edit Descriptive Flexfield page, specify the prompt, whether the segment should be displayed, and whether a value is required.

As explained in Task: Define Validation Rules for Extensible Flexfield Custom Attributes in Section 5.3, "Planning Your Flexfields," you do not need to specify a value set if the set of valid values is the same as the set of context values. If you need to associate a context with a value set, then the value set must be an independent or table value set. The data type must be Character and the maximum length of the values being stored must not be larger than the context's column length.

Note: You cannot assign a table value set to a context segment if the WHERE clause uses VALUESET.*value_set_code* or SEGMENT.*segment_code* bind variables.

Using the list of valid context values that you prepared in the planning stage, access the Manage Contexts page and create a context for each value. Create the context-sensitive segments for each context and provide the identifying information, the column assignment, how the value should be validated, and the display properties.

Note: The context and segment codes are used in the flexfield's element in the XML schema for web services. The use of leading numeric characters, spaces, underscores, and multibyte characters, which are all encoded in XML schemas, make the codes in the schema element difficult to read.

Tip: When you create segments for your attributes, you specify the sequence number for the segment. This sequence affects the order in which the attribute is displayed on the page. Before you begin creating segments, you might want to plan how you will number sequences. Consider numbering the segments in multiples, such as 4, 5, or 10, to make it easy to insert new attributes.

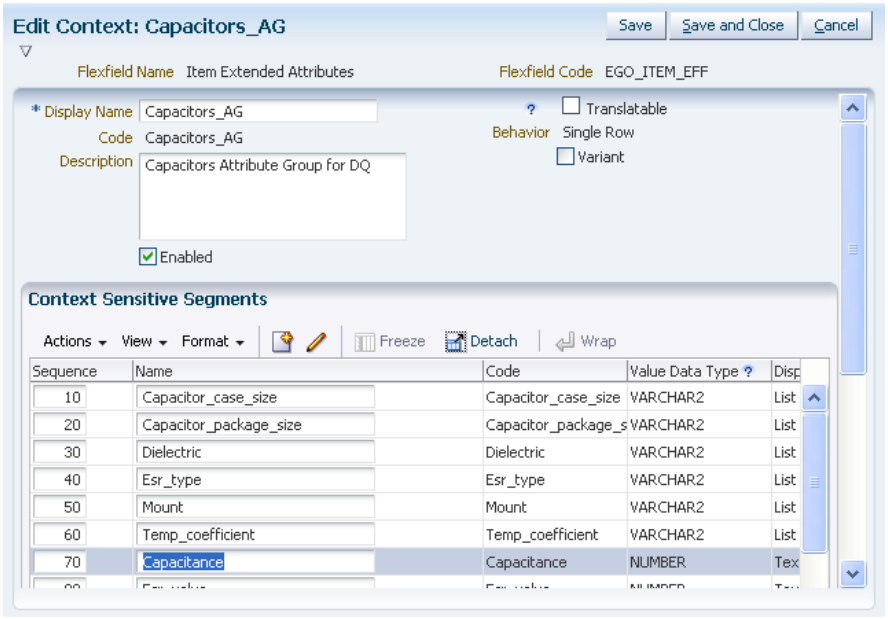
5.5.2 Configuring Extensible Flexfields

Use the Manage Extensible Flexfields task shown in Figure 5–14 to configure an extensible flexfield.

You configure an extensible flexfield by defining its contexts and associated context-sensitive segments and usages, associating the context with categories and pages, and defining the display properties and value validation for each context-sensitive segment.

If the Oracle Fusion applications are running in different locales, you might want to provide different translations for the translatable text, such as prompts and descriptions. To create the translations, log in with each of the different locales (or choose **Personalization** from the **Set Preferences** menu in the global area to set the locale) and change the text to the translated text for that locale. For more information, see Section 16.5, "Translating Flexfield and Value Set Configurations."

Figure 5–14 Edit Context Page in the Manage Extensible Flexfields Task



For more information, see the "Manage Extensible Flexfields" section in the *Oracle Fusion Applications Common Implementation Guide*.

Before you begin

You will need to do the following before you can begin configuring an extensible flexfield:

- Plan your flexfield configuration as described in Section 5.3, "Planning Your Flexfields."

- Create the required value sets as described in Section 5.4, "Creating Custom Value Sets."
- Compile a list of the UI pages that are affected by the extensible flexfield as well as other artifacts in the Oracle Fusion Middleware technology stack. See Section 5.8, "Integrating Custom Attributes" for more information about how flexfield configuration affects these artifacts. Using this list, plan how you will integrate and test your flexfield configuration
- Ensure that you have the necessary function and data security privileges to access the Manage Extensible Flexfields task. Contact your security administrator for details.
- Access the Manage Extensible Flexfields task by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task.

Task: Configure the Context

Using the list of contexts that you prepared in the planning stage, access the Manage Contexts page and create the desired contexts. Create the context-sensitive segments for each context and provide the identifying information, the column assignment, how the value should be validated, the initial default value, and the display properties.

Some extensible flexfields have a Translatable option. If the context will be used to store free-form user-entered text in the language of the user's locale, and if different translations of that text can be stored for other languages, then select **Translatable**. If you select **Translatable**, then the context can have only format-only validated segments.

Note: The context and segment codes are used in the flexfield's element in the XML schema for web services. The use of leading numeric characters, spaces, underscores, and multibyte characters, which are all encoded in XML schemas, make the codes in the schema element difficult to read.

Tip: When you create segments for your attributes, you specify the sequence number for the segment. This sequence affects the order in which the attribute is displayed on the page. Consider numbering the segments in multiples, such as 4, 5, or 10, to make it easy to insert new attributes.

Task: Configure Context Usages

In Task: Find Extensible Flexfields on a Page in Section 5.2, "Finding the Flexfields on a Page," you learned how to identify the flexfield usage for which you are adding attributes. You can associate a context with several usages, but ensure that the **Associate** checkbox is selected for the usage that you identified in that task. Consult the product-specific documentation in Oracle Fusion Applications Help to understand the purpose of each usage.

Task: Configure Categories and Category Details

While most extensible flexfields have a single category, some extensible flexfields have multiple categories. When an extensible flexfield has multiple categories you can further group the contexts into categories. Some flexfields provide an activity or task for creating your own categories. Consult the product-specific documentation in

Oracle Fusion Applications Help to determine whether you can create categories for the flexfield.

Task: Associate Contexts with a Category

Most extensible flexfields have a single category that contains all the flexfield's contexts. Some flexfields enable you to work with multiple categories. To associate contexts with a category, select the category on the Edit Extensible Flexfield page. Then, in the Associated Contexts tab, add the desired contexts to the selected category.

Task: Create Logical Pages for a Category

Using the information from your plan, group the contexts into logical pages and determine the sequence in which the logical pages should appear. To build the logical pages, select the category on the Edit Extensible Flexfield page. Then, in the Pages tab, add the desired contexts and specify their sequence.

5.6 Validating Flexfield Configurations

You can verify that a flexfield's metadata is complete and correctly configured by choosing **Actions** then **Validate Metadata** in either the Maintain Extensible Flexfields task or the Maintain Descriptive Flexfields task. The validations that this action performs are the same validations that are performed when you deploy a flexfield.

5.7 Deploying Flexfield Configurations

You must deploy a flexfield before you can access the custom attributes from the user interface and other parts of the Oracle Fusion Middleware technology stack. You use the appropriate flexfield management task — Manage Descriptive Flexfields or Manage Extensible Flexfields — to deploy a flexfield.

When you deploy descriptive or extensible flexfields, the following occurs:

- The deployment process validates the flexfield metadata. If any errors are found, then a popup window is displayed with a list of the errors that were encountered, and the flexfield is not deployed. Note that the flexfield's status is not changed if metadata errors are encountered.
- The deployment process generates ADF Business Components and ADF Faces runtime artifacts that are based on the flexfield configurations.
- The deployment process adds the custom attributes to the Web Service Description Language (WSDL) schemas that are exposed by Oracle ADF services and that are used by SOA composites. For information about SOA, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
- If the flexfield is enabled for business intelligence, the deployment process redeploys the flexfield's business intelligence artifacts.
- The web services that expose the business object's data also expose the business object's flexfield segment data.
- End users see the new custom attributes the next time they log in to the application. The custom attributes will appear on all user interface pages that contain the flexfield, and in the search screens and Excel worksheets with ADF Desktop Integration in which they have been integrated.

If you want to test the flexfield configuration before deploying it to the full test environment, then you can deploy the flexfield to a flexfield sandbox. The changes that you deploy to a sandbox are isolated from the full test environment and can be

seen only by those who make the flexfield sandbox active in their session. After you are satisfied with the changes in the sandbox, you can deploy the changes to the full test environment.

After deployment, the custom attributes are available for incorporating into the SOA infrastructure, such as business process and business rule integration. For example, you can now write business rules that depend on the custom attributes.

Descriptive flexfields that are enabled for business intelligence are available for integrating with business intelligence (BI) technology, such as Oracle BI Enterprise Edition (Oracle BI EE) and Oracle Essbase.

For more information about flexfield deployment and flexfield sandboxes, see the "Flexfield Deployment" section in the *Oracle Fusion Applications Common Implementation Guide*. For more information about integrating the deployed flexfield into the Oracle Fusion Middleware technology stack, see Section 5.8, "Integrating Custom Attributes."

Before you begin

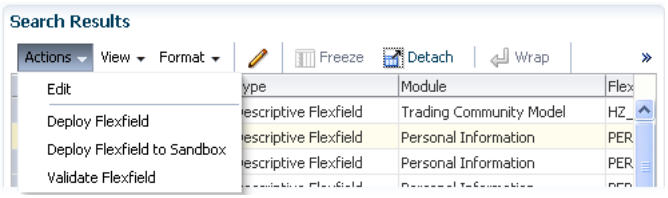
You will need to do the following before you can begin deploying a descriptive or extensible flexfield:

- Configure the flexfield as described in Section 5.5, "Configuring Flexfields."
- Ensure that you have the necessary function and data security privileges to access the flexfield management task. Contact your security administrator for details.
- Access the appropriate flexfield management task — Manage Descriptive Flexfields or Manage Extensible Flexfields — by choosing **Setup and Maintenance** from the **Administration** menu in the global area of Oracle Fusion Applications and searching for the task.
- Use the Search region to access the desired flexfield.

Task: Deploy a Flexfield to a Sandbox

If you want to test the flexfield configuration before deploying it to the production environment, then you can deploy the flexfield to a sandbox. To deploy a flexfield to a sandbox, select the flexfield in the Search Results region, and choose **Deploy Flexfield to Sandbox** from the **Actions** menu, as shown in Figure 5–15. A progress window shows when the process has completed and the completion status.

Figure 5–15 Deploying a Flexfield to a Sandbox



The sandbox to which you deploy the flexfield is different from the standard sandboxes that are described in Section 2.2, "Using the Sandbox Manager." Each flexfield has its own sandbox. You do not need to create a flexfield sandbox because the deployment process manages flexfield sandbox creation. When you deploy to the flexfield sandbox, a dialog box shows the name of the flexfield sandbox, and that flexfield sandbox is set as your current active sandbox.

Note: When you deploy to the flexfield sandbox, the `CreatedBy` value is set to `FlexfieldDeployment`.

When you next log in to the application, you can see the updated flexfield configurations (you must log out and log back in to see the changes). The Oracle Fusion Applications global area displays your current session sandbox. When you hover over the sandbox name, the sandbox details appear. To exit a sandbox, hover over the sandbox name and click **Exit Sandbox**. To view additional details about the sandbox, hover over the sandbox name and click **More Details**.

If you want to make an existing flexfield sandbox active for your session, then access the flexfield sandbox from the Manage Sandboxes page, select the flexfield sandbox, and click **Set as Active**. For information about accessing this page, see Section 2.2, "Using the Sandbox Manager."

After you are satisfied with the changes, you can deploy the flexfield to the full test environment as described in Task: Deploy a Flexfield to the Full Test Environment.

Task: Deploy a Flexfield to the Full Test Environment

To deploy a flexfield to the full test environment, select the flexfield in the Search Results region, and select **Deploy Flexfield** from the **Actions** menu.

A progress window shows when the process has completed and the completion status. If a sandbox exists for the flexfield, then this sandbox is deleted automatically after the flexfield successfully deploys to the full test environment.

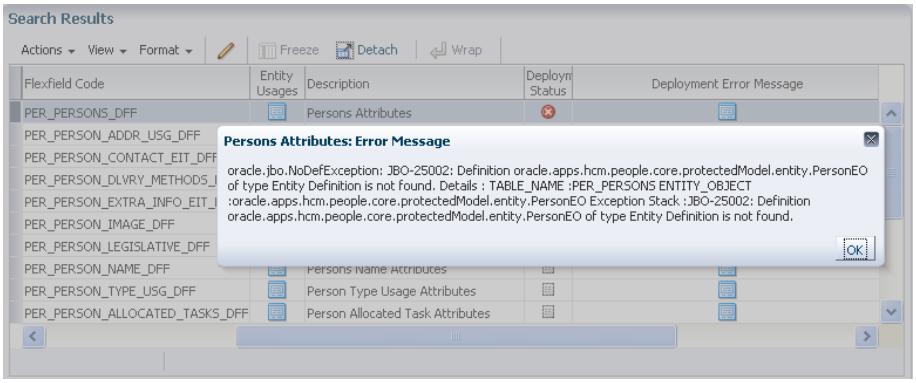
When you next log in to the application, you can see the updated flexfield configurations (you must log out and log back in to see the changes).

Task: Check Deployment Status

The Deployment Status column in the Search Results region shows the current status for each flexfield.

If a deployment error occurs, then the Deployment Error Message column in the Search Results region provides details about the error, as shown in Figure 5–16.

Figure 5–16 Deployment Error Message



5.8 Integrating Custom Attributes

After you have deployed a flexfield, you can begin incorporating its custom attributes into the Oracle Fusion Middleware technology stack.

For information about the technology stack, see the "Oracle Fusion Middleware Components" chapter in the *Oracle Fusion Applications Concepts Guide*.

Task: Customize Flexfield Pages

After you deploy an extensible or descriptive flexfield, the new attributes appear on all pages that contain the flexfield. You can use Page Composer to configure the custom attribute properties on a page-by-page basis. For example, you can hide some custom attributes. For more information, see Chapter 3, "Customizing Existing Pages."

Note: The custom attributes appear only on pages that include the flexfield. For information about adding flexfields to a page, see the "Adding Descriptive Flexfield UI Components to a Page" section or the "Employing Extensible Flexfields on an Application Page" section in the *Oracle Fusion Applications Developer's Guide*.

Task: Incorporate Custom Attributes into Oracle Business Intelligence

Some descriptive flexfields are enabled for incorporating into Oracle Business Intelligence data structures. Consult the product-specific documentation from Oracle Enterprise Repository for Oracle Fusion Applications for information about which flexfields are enabled for Oracle Business Intelligence.

If a descriptive flexfield is enabled for Oracle Business Intelligence, the Manage Descriptive Flexfields task displays a **BI Enabled** checkbox for each global, context, and context-sensitive segment. Select a segment's **BI Enabled** checkbox to specify that the segment is available for use in Oracle Business Intelligence.

When you deploy a flexfield with BI-enabled segments, the deployment process generates a set of *flattened* ADF Business Components in addition to the normal ADF Business Components and ADF Faces runtime artifacts that are generated during deployment. A flattened business component includes a single attribute for the context segment if it is BI-enabled, as well as one attribute for each BI-enabled global segment. A flattened business components includes a separate attribute for each BI-enabled context-sensitive segment for each context value. Context-sensitive segments are not equalized across context values.

After you deploy a business intelligence enabled flexfield, you must import the flexfield changes into the Oracle Business Intelligence repository to make use of the newly flattened components in business intelligence and then propagate the flex object changes.

Note: When you import the metadata into the Oracle Business Intelligence repository, you must do so as the `FUSION_APPS_BI_APPID` user.

Tip: When you import a flexfield into the Oracle Business Intelligence repository, you see both *name*_ and *name*_c attributes for each segment, along with some other optional attributes. The *name*_ attribute contains the value. The *name*_c attribute contains the code of the value set that the value comes from, and is used for linking to "value dimension." You must import both attributes.

For more information about importing and propagating your flexfield changes, see Task: Configuring Descriptive Flexfields and Key Flexfields for Oracle Business

Intelligence in Section 8.3.3, "Customizing the Oracle BI Repository (RPD)." For information about adding flexfields to an Oracle Business Intelligence Publisher data model, see the "Adding Flexfields" chapter in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)*.

Task: Incorporate Custom Attributes into Web Services and SOA Infrastructure

When an extensible or descriptive flexfield is deployed, the deployment process regenerates the XML schema definition (XSD), which makes the custom attributes available to web services and the SOA infrastructure.

After deploying a flexfield configuration, you must synchronize the updated XSD files in the SOA MDS repositories for each SOA application. For more information, see Task: Synchronizing Customized Flexfields in the SOA MDS Repository in Section 12.2, "Customizing SOA Composites."

For information about extending or customizing Business Process Execution Language (BPEL) processes, see Part "Using the BPEL Process Service Component" of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

For more information about service-oriented architecture, web services, BPEL processes, and business rules, see the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

6

Customizing the Navigator Menu

This chapter describes how to customize the navigator menu in Oracle Fusion Applications by using the Manage Menu Customizations task in the Setup and Maintenance work area to add and delete menu groups and items.

This chapter includes the following sections:

6.1 About Navigator Menu Configuration

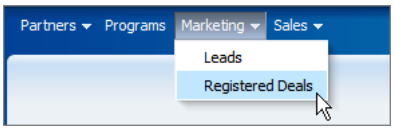
The *navigator menu* is the global menu that is accessible from the Oracle Fusion Applications global area. It allows an end user to navigate directly to pages inside of Oracle Fusion Applications as well as to outside web pages. The menu is composed of links (*items*) that are organized in a hierarchy of *groups*.

You can customize the navigator menu to address needs that are specific to your organization. For example, you might want to add specialized groupings for cross-functional teams or add links to web pages or external applications. You can add groups and links to the navigator menu, as well as hide and show them.

The navigator menu typically appears when the end user clicks the Navigator link, as shown in Figure 6–1. However, in some applications the page template can be customized to present the top level groups and items as pull-down buttons, as shown in Figure 6–2, in place of the single Navigator link. For information about how to display the navigator menu as a series of pull-down buttons see "Rendering the Navigator Menu as Pull-down Buttons" in the *Oracle Fusion Applications Developer's Guide*.

Figure 6–1 Navigator Menu Example — Navigator Link

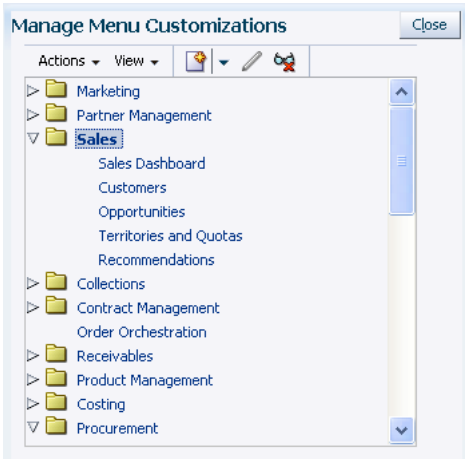
Figure 6–2 Navigator Menu Example — Navigator Pull-down Buttons



In a multi-tenant environment, you customize the navigator menu at the site level and your changes affect the tenant at that site level. Otherwise, you customize the navigator menu at the site level and your customizations affect all end users.

You use the Manage Menu Customizations task to customize the menu. This task is available from the Setup and Maintenance work area, which is accessible from the Administration menu in the Oracle Fusion Applications global area. The Manage Menu Customizations page displays the menu groups as expandable nodes, as shown in Figure 6–3, with which you can traverse the menu hierarchy.

Figure 6–3 Expandable Group Nodes in Manage Menu Customizations Page



Note: Not all Oracle Fusion Applications pages appear in the navigator menu, as some pages are accessible from a work area or from other links in the global area such as the Home link.

You can also customize the Navigator menu from Oracle JDeveloper. For more information, see Section 11.11, "Customizing Menus"

6.1.1 What You Can Do with the Navigator Menu at Runtime

If you have the required privileges, you can perform the following tasks to customize the menu:

- Add and delete custom groups
- Edit any group
- Add and delete custom items
- Edit any item
- Specify navigation for an item
  - Specify navigation to a UI Shell page in an Oracle Fusion application
  - Specify navigation to an external web page
- Hide or show groups and items

You can also localize your navigator menu customizations. For more information, see Section 16.4, "Translating Navigator Menu Customizations."

6.1.2 What You Cannot Do with the Navigator Menu at Runtime

You cannot make the following menu customizations at runtime:

- You cannot add menu items (links) as top level nodes. You only can add nodes to the groups in the top level and to subgroups.
- You cannot delete nodes that are delivered with the product.
- You cannot move nodes. Instead, you must duplicate the node and hide the original node.

Note: While you can customize the global navigator menu at runtime, you cannot customize the global home menu or the global preferences menu at runtime.

For information about customizing the navigator and home menus using Oracle JDeveloper, see Section 11.11, "Customizing Menus"

6.1.3 Before You Begin Customizing the Navigator Menu

Before you customize the navigator menu, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can begin customizing the navigator menu:

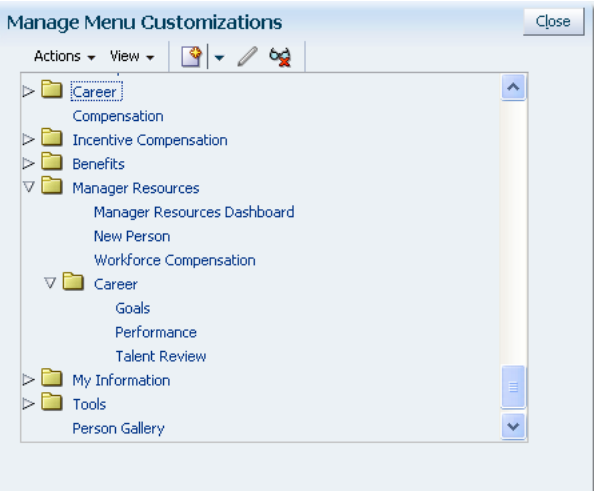
- If you are making minor changes, you can hide the changes until you have completed your customizations. However, if you are making more than minor changes, you might want to instead create a sandbox. For more information see Section 2.2, "Using the Sandbox Manager."

- You must have the necessary function and data security privileges to access the Manage Menu Customizations task. Contact your security administrator for details.
- Open the Manage Menu Customizations task. To access this task, choose **Setup and Maintenance** from the Administration menu in the Oracle Fusion Applications global area and search for the task.

6.2 Adding Groups

You arrange the menu by building a hierarchy of nested groups. For example, in Figure 6–4 the Manager Resources group contains the Career group.

Figure 6–4 Menu Groups and Sub Groups



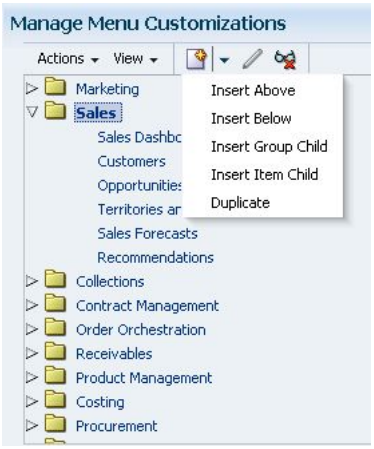
Task: Navigate the Menu Hierarchy

You can use the View menu to expand or collapse a group of nodes, scroll to the first group, or scroll to the last group in the navigation tree. You can also right-click a node and access similar actions to facilitate tree navigation.

Task: Add a Group

As shown in Figure 6–5, you can insert a group above or below a peer group or insert a group child. You edit a group by defining a label and specifying whether the group should be rendered. You typically hide the group until all changes have been completed.

Figure 6–5 Action Menu



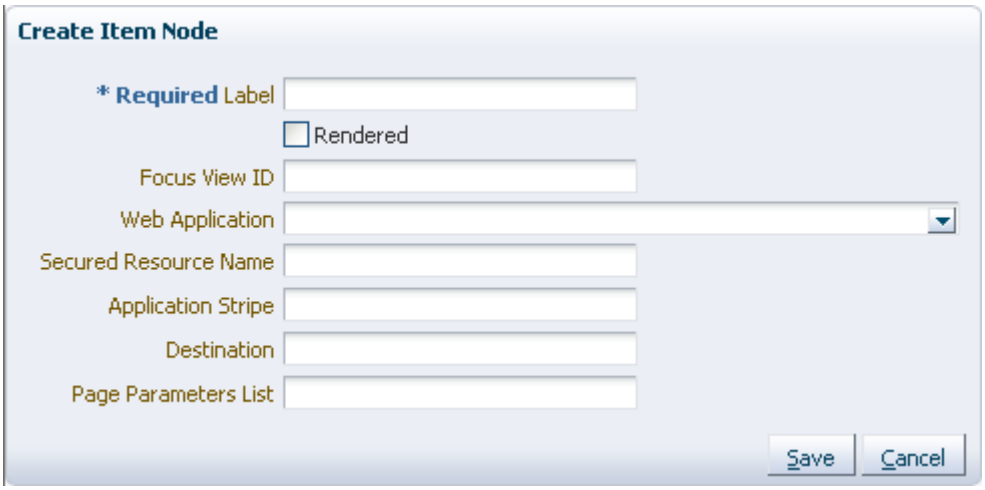
6.3 Adding Items

Navigator menu items are URL links. There are two types of links — links to UI Shell pages in Oracle Fusion applications and links to external applications and web sites. Menu items can be added to the top level groups and their subgroups. Note that you cannot add menu items as top level nodes.

Task: Adding an Item

To add an item, you navigate to the item's group and insert the item above or below another item, as shown in Figure 6–5. You can also choose to duplicate an existing item. You must supply the menu label and either link to a UI Shell page or link to an external web site or application, as described in the following tasks. The Create Item Node page is shown in Figure 6–6.

Figure 6–6 Create Item Node Page



Task: Linking to a UI Shell Page

If the new item points to a UI Shell page in an application, you must provide the name of the web application and the view ID of the target page. The quickest way to obtain the web application name and view ID is to copy them from an existing menu item that links to the same page. Otherwise, the web application name is the same as the

context root for the application, and the view ID can be obtained from the `id` attribute for the page's `<view>` tag in the product's `public_html/WEB-INF/adfc-config.xml` file.

If you want to secure access to the target UI Shell page from the menu item, you will need to provide the name of the secured resource and the name of the policy store's application stripe. When an end user clicks the link, Oracle Fusion Applications checks the secured resource and Lightweight Directory Access Protocol (LDAP) policy store to determine whether the user has permission to view the page.

If there is another menu item that points to the same page, you can get the secured resource name and application stripe from that item. Otherwise, the application stripe can be obtained from the `jps.policystore.applicationid` parameter in the application's `weblogic-application.xml` file. Examples of application stripes are `crm`, `fscm`, and `hcm`. The secured resource name is the name of the web page's page definition file. By default, the page definition files are located in the `view.PageDefs` package in the Application Sources directory of the view project. If the corresponding JSF page is saved to a directory other than the default (`public_html`), or to a subdirectory of the default, then the page definition will also be saved to a package of the same name. An example of a secured resource name is `oracle.apps.view.pageDefs.CaseList_Form_Attach_UIShellPagePageDef`.

A UI Shell page might take parameters and display or act differently based on the parameters that are passed in. For example if accessing a page from one group in the menu hierarchy, the parameter might be set to `status="Open"` and if accessing the page from a different group, the parameter might be set to `status="Closed"`. If the page takes parameters, you can use the **Page Parameters List** text box to provide a semicolon-delimited string of name value pairs, such as `org=m1;context=s1`. You can use expression language (EL) to specify the parameters. If the EL evaluates to an Object, the `toString` value of that Object is passed as the value of the parameter.

Note: Do not enter a value in the **Destination** text box. If a destination is provided, the menu item is treated as a link to an external web page and the target view ID and web application values are ignored.

Task: Linking to an External Web Site or Application

You can link a menu item to an external web site or application. Clicking on the menu item displays the target in a new browser window or tab, depending on the browser configuration.

To link to an external web site or application, provide the URL in the **Destination** text box and provide the menu label. Do not enter information in any other fields. If you need to pass parameters, put the parameters in the URL.

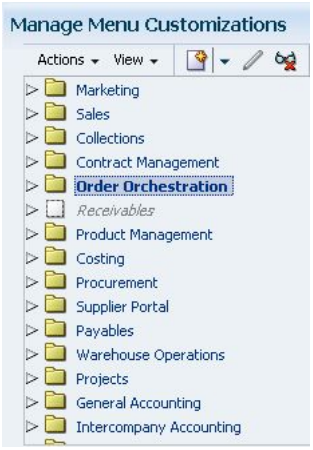
6.4 Hiding and Showing Nodes

While you are creating or working with a group or item, you might want to prevent end users from accessing the node. You can hide the group or item while you are working with it, and then show the node when you have completed the task.

Tip: For major changes that need to be tested and approved, you might want to use the sandbox manager instead of hiding and showing nodes. For more information, see Section 2.2, "Using the Sandbox Manager."

The Manage Menu Customizations page shows all nodes. The nodes that appear in italics are hidden from end users, as shown in Figure 6–7.

Figure 6–7 Hidden Node Shown in Italics



Task: Hiding or Showing a Node

When you add a node, you can select the Rendered checkbox to display the node, or clear the checkbox to hide it. You can edit the node later to change how it is rendered.

7

Customizing and Extending BPMN Processes

This chapter describes how to use Oracle Business Process Composer to customize and extend Business Process Modeling Notation (BPMN) processes. Several Oracle Fusion applications use BPMN processes to define process flows within the application. This chapter also describes how to edit BPMN processes by creating and modifying BPM projects based on project templates and deploying those projects to runtime.

This chapter includes the following sections:

7.1 About Customizing BPMN Processes

The Oracle Fusion Customer Relationship Management (CRM) and Oracle Fusion Human Capital Management (HCM) product families of Oracle Fusion Applications use Business Process Management Notation (BPMN) processes to define some of the process flows used within their applications.

BPMN is a standard notation for modeling the behavior of business processes. It enables process analysts to create graphical models of a business process. The Oracle Business Process Management Suite provides an environment for implementing BPMN processes by enabling process developers to integrate them with other systems and services. The Oracle BPM Suite also provides a runtime environment for integrating the running processes within Oracle Fusion Applications. For general information on BPMN, see the "Modeling Business Processes with Oracle BPM" chapter in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

Business Process Composer enables you to customize the BPMN processes used within Oracle Fusion Customer Relationship Management and Oracle Fusion Human Capital Management. To customize these processes you must create, modify and deploy Oracle BPM projects created from a project templates. BPMN processes are contained within a BPM project. BPM projects contain all of the resources required for a functioning BPM application, including BPMN processes and SOA artifacts such as business rules and human tasks.

Oracle Fusion applications provide default project templates that you can use to create new process flows. See the product-specific documentation in Oracle Fusion Applications Help for more information on the default BPM project templates provided by Oracle Fusion Applications.

For general information on working with project templates, see the "Introduction to Project Templates" section in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

A BPM project template is pre-populated with all of the required resources for implementing BPMN processes within an Oracle Fusion application. This includes the BPMN processes that define the process flow as well as the necessary technical components that enable the processes to communicate with other parts of the application.

After customizing a BPM project, you can deploy it to runtime in one of several ways. See Section 7.5, "Deploying an Oracle BPM Project" for more information.

The technical components contained within a BPM project are called the business catalog. The business catalog contains various reusable services that a BPMN process can use to connect to other components of the application, including other processes, systems, and databases.

The business catalog contains the following reusable components:

Table 7–1 Business Catalog Components Available in Business Process Composer

Business Catalog Component	Description
Services	Services are used to connect a BPMN process with other processes, systems, and services, including BPEL processes, databases. See the "How to Create New Services in the Business Catalog" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for more information.
Human tasks	Human tasks enable you to define how end users interact with your BPMN processes. Human tasks are implemented in a BPMN process using the user task. See the "Adding User Interaction to Your Process" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management.*
Business rules	Oracle business rules are statements that describe business policies or describe key business decisions. See the "Introduction to the Business Rules Task" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on implementing business rules within a BPMN process.

7.1.1 Oracle Tools for Customizing and Extending BPMN Processes

Oracle Fusion applications provide multiple tools for customizing and extending BPMN processes. These tools are described in Table 7–2.

Table 7–2 Oracle Tools for Customizing and Extending BPMN Processes

This tool...	Enables you to...
Business Process Composer	Customize BPMN processes by creating and deploying BPM projects based on project templates. This functionality is described in the current chapter.
Oracle BPM Studio (JDeveloper)	Customize project templates. See Chapter 13, "Customizing and Extending Oracle BPM Project Templates."
Oracle SOA Composer	Customize business rules, domain value maps, and approval configuration and assignment rules at runtime. See Section 12.2, "Customizing SOA Composites" for more information. These customizations are performed directly on a running application. They do not require redeployment of the BPM project containing the BPMN process.
Oracle BPM Worklist	Customize approval configuration and assignment rules. See Section 12.2, "Customizing SOA Composites" for more information. These customizations are performed directly to a running application. They do not require redeployment of the BPM project containing the BPMN process.

7.1.2 What You Can Do with BPMN Processes at Runtime

There are two types of runtime customization that you can make to BPMN processes. These are described in the following sections.

7.1.2.1 What You Can Customize Using Oracle SOA Composer and Oracle BPM Worklist

BPMN processes utilize multiple SOA components, including business rules and approval workflow. Using the Oracle SOA Composer and Oracle BPM Worklist, you can customize the following components used by the BPMN processes of a running Oracle Fusion application:

- Oracle business rules
- Domain value maps
- Approval assignment rules in human workflows such as customizing the approval flow for a specific customer.

These customizations can be made directly to a running Oracle Fusion application without having to redeploy the BPM project. See Section 12.2, "Customizing SOA Composites" for information on using Oracle SOA Composer or Oracle BPM Worklist to make these customizations.

> **Note:** Any changes you make to the deployed, running BPMN processes of an Oracle Fusion Customer Relationship Management application will not be preserved if you later redeploy the BPM project that contains them.

7.1.2.2 What You Can Customize Using Business Process Composer

Using Business Process Composer, you can make the following customizations to a BPM project created from a project template:

- Customize an existing BPMN process.
- Create new BPMN processes.

- Create simple data objects.
- Create and modify some business catalog components.

  Table 7–3 lists which business catalog components can be customized or created using Business Process Composer.

Table 7–3 List of Business Catalog Components

Business Catalog Component	Can be created using Business Process Composer?	Can be customized using Business Process Composer?
Business rules	No	Yes
Human tasks	Yes You can create human tasks using Business Process Composer, however not all functionality of a human task can be customized.	Yes
Services	Yes	Yes

See Section 7.2, "Creating an Oracle BPM Project" for information on creating a new BPM project based on a project template.

Changes to the business catalog that cannot be made using Business Process Composer must be made to the project template using Oracle BPM Studio. See Section 7.1.3, "What You Cannot Do with BPMN Processes at Runtime" for more information.

7.1.3 What You Cannot Do with BPMN Processes at Runtime

Often, it is necessary to make changes to a BPMN process that cannot be performed using Business Process Composer. The following tasks must be performed using JDeveloper:

- Modify project templates
- Create or customize some business catalog components, including:
  - Configuring advanced properties of web services
  - Creating new adapters and mediators
  - Creating new business rules
  - Creating or customizing errors
  - Configuring advanced features of human tasks, including:
    * Complex assignment support
    * Auto-generated task flow
    * Business rules within human tasks
- Create complex data objects

To create or customize these components, process developers must modify the project template and republish it to the Oracle BPM repository. After a project template is revised, you can use Business Process Composer to create and deploy BPM projects. See Chapter 13, "Customizing and Extending Oracle BPM Project Templates" for more information.

7.1.4 Before You Begin Customizing BPMN Processes

Before customizing BPMN processes using Business Process Composer, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications."

You should also understand the typical workflows for customization and extensibility, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You should be familiar with how to model business processes using the Business Process Modeling and Notation (BPMN) standard. See the "Modeling Business Processes with Oracle BPM" chapter in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

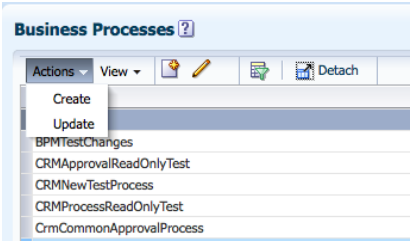
To create and deploy projects based on project templates, you must be granted the correct permissions by your security administrator. See the "Performing Administrative Tasks" chapter in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for more information.

How you access Business Process Composer depends on which Oracle Fusion application you are using:

- To access Business Process Composer from Oracle Fusion Human Capital Management, access the **Business Process Composer** task by choosing **Setup and Maintenance** from the Administration menu in the global area of Oracle Fusion Applications and searching for the task.
- To access Business Process Composer from Oracle Fusion Customer Relationship Management:

  1. Access CRM Application Composer
     - To access CRM Application Composer, in the **Navigator** menu, choose **Application Composer**.
     - To edit or create business processes, you will need the correct privileges. Please contact your security administrator for details.
  2. Access Business Process Composer
     - To access Business Process Composer from CRM Application Composer, click **Business Processes** as shown in Figure 7–1.

Figure 7–1 Actions Menu of the CRM Application Composer



After selecting one of the above, the Business Process Composer login screen appears. For more information, see the "Signing On to Business Process Composer" section in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

- To create a new business process select **Create** from the **Actions** menu.
- To modify an existing business process select a process from the list, then select **Update** from the **Actions** menu.

7.2 Creating an Oracle BPM Project

You can use Business Process Composer to customize or extend BPMN processes. BPMN processes are contained within an Oracle BPM project. After launching Business Process Composer the first step in modifying a BPMN process is to create a new project based on a project template or to open an existing BPM project.

Task: Create a New BPM Project Based on a Project Template

Oracle Fusion applications do not ship with running BPMN processes out-of-the-box. In order to integrate BPMN processes within Oracle Fusion applications, you must create a new BPM project based on a project template, deploy the project to runtime, then configure the Oracle Fusion application to use the BPMN processes of the deployed BPM project.

Business Process Composer enables you to create new BPM projects based on project templates. For information on creating a new BPM project, see the "How to Create a New Project From a Project Template" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

Refer to the product-specific documentation in Oracle Fusion Applications Help for a list of the default BPM project templates provided by Oracle Fusion Applications. By default, project templates are stored in the Templates folder of the BPM repository.

Task: Open an Existing BPM Project

If you have already created a new BPM project based on a template, you can continue to customize the project before deploying it to runtime.

For information on opening an existing BPM project, see the "How to Open a Project Using the Project Browser" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

7.3 Customizing BPMN Processes

After creating a new BPM project based on a project template, you can modify the BPMN processes within the project. Additionally, you can customize the business catalog components within the BPM project.

Task: Open a BPMN Process

See the "How to Open a Business Process" section of the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on opening a BPMN process.

Task: Customize a BPMN Process

You can alter the flow of your BPMN process by adding, removing, or modifying BPMN flow objects.

See the "Working with Flow Elements" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for more information on using the process editor to modify BPMN processes.

See the "Modeling Business Processes with Oracle BPM" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for specific information on BPMN flow objects.

Task: Assign Business Catalog Components to a BPMN Flow Object

You can use Business Process Composer to assign reusable services from the business catalog to different BPMN flow objects.

The business catalog components and their corresponding flow objects are shown in Table 7–4.

Table 7–4 BPMN Flow Objects and Their Corresponding Services

BPMN Flow Object	Business Catalog Component
User task	Human tasks
Service task	Services, including web services and adapters
Business rules task	Business rules

See the "The Business Catalog" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for more information on working with reusable services and the business catalog.

Task: Add Milestones to the Activity Guide

The activity guide of a BPM project defines a set of milestones. Each BPM project contains one activity guide. An activity guide can contain multiple milestones.

A milestone is a specific set of tasks that the process participant has to complete. A milestone is complete when the user successfully runs a specific set of tasks in the milestone.

Each milestone is a defined by a set of human workflow tasks. Each human workflow task is itself a task flow that may require the collaboration of multiple participants in various roles.

See the "Using Guided Business Processes to Set Project Milestones" section in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management.* for information on creating milestones.

Task: Customize Business Rules

Business rules enable dynamic decisions to be made at runtime that allow you to automate policies, computations, and reasoning while separating rule logic from underlying application code.

> **Note:** You can use Business Process Composer to modify business rules within an Oracle BPM project. These changes are made to the runtime application when the project is deployed.
>
> You can use the Oracle SOA Composer to make runtime changes directly to the runtime environment, without deploying a project. See Section 12.2, "Customizing SOA Composites" for more information making runtime changes to a BPMN process.

For more information about customizing rules using Business Process Composer, see the "Using Oracle Business Rules" chapter in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

Task: Create or Customize Human Tasks

Human tasks enable you to integrate human interaction with connectivity to systems and services as part of an end-to-end process flow. Human tasks are responsible for handling all interactions with the users or groups participating in the business process.

Using Business Process Composer, you can create and customize human tasks. See the "Working with Human Tasks" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on creating and configuring human tasks.

Task: Customize Expressions

Expressions are used to evaluate the data used within your process. Different flow objects use expressions to determine which path within the process to follow.

Expressions are used to configure the following BPMN elements:

- Conditional Sequence Flows
- Complex Gateways
- Timer Events
- Data Associations
- Loop Markers
- Multi-Instance Markers
- User Task Advanced Properties

You can customize the expressions used within a BPMN process using Business Process Composer.

See the "Working with Expressions" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

7.4 Saving an Oracle BPM Project to the BPM Repository

You can save the changes you make to a BPM project to the BPM repository. In addition to saving your work in progress, saving a project in the repository enables you to share BPM projects with process developers using Business Process Composer and Oracle BPM Studio.

The BPM repository can also be used to share project templates created in Oracle BPM Studio. In the BPM repository, projects and project templates are stored in the following default folder:

Table 7–5 Default Location of BPM Projects and Project Templates in the BPM Repository

Type	Location
BPM Projects	/bpm/drafts
BPM Project Templates	/bpm/templates

Within these default folders you can create additional subfolders to organize your projects and project templates.

Task: Save a Project

See the "How to Save Changes to a Project" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on how to save a project to the BPM repository.

7.5 Deploying an Oracle BPM Project

After creating a new BPM project, you must deploy it to runtime in order for the BPMN processes contained within it to be accessible to Oracle Fusion applications.

After customizing a project and publishing it, you must deploy the project to runtime. After the project is deployed, the BPMN processes within it are accessible to the Oracle Fusion applications that implement them.

Task: Deploy a BPM Project

There are three methods of deploying a BPM project to Oracle Fusion applications using Business Process Composer. The specific method you use depends on whether you are in a production or test environment:

- Using a SAR file

  This is the recommended deployment method if you are deploying a BPM project to a production environment. A SAR file is an archive of a SOA composite application that, like a BPM project, contains all of the required resources of a deployable application.

  You can use Business Process Composer to export a BPM project as a SAR file. See the "How to Deploy a project to runtime" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*.

  After the project is exported as a SAR file, your system administrator must deploy it to runtime using Oracle Enterprise Manager.

- Using a deployment plan

  You can use Business Process Composer to generate a deployment plan for your project. Like a SAR file, system administrators can use a deployment plan to deploy a BPM project to runtime.

  See the "How to Generate a Deployment Plan" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on generating a deployment plan using Business Process Composer.

- Directly from Business Process Composer

  You can deploy a BPM project to runtime directly from Business Process Composer. See the "How to Deploy a Project to Run Time" section in the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for information on deploying a BPM project to runtime.

  In order to deploy project using Business Process Composer you must be granted the correct permission by your system administrator.

Note: The initial deployment of a BPM project cannot be performed using Business Process Composer. The first time you deploy a project, you should use one of the other deployment procedures. This is because projects stored in the BPM repository contain abstract references to application resources. When deploying a BPM project using a SAR file, these abstract references are modified to point directly to Oracle Fusion application resources.

When deploying a new version of a BPM project, any runtime changes you made to the project using Oracle SOA Composer or Oracle BPM Worklist will not be preserved after redeploying the BPM project.

You may want to deploy a BPM project using the same name as the project and project template to make it easier to associate the deployed project with its design time version

Task: Configure Oracle Fusion Applications to Use BPMN Processes

After deploying a BPM project, you must configure the Oracle Fusion application to use the BPMN processes within the application. See Section 7.6, "Configuring Oracle Fusion Applications to Use BPMN Processes" for more information.

7.6 Configuring Oracle Fusion Applications to Use BPMN Processes

After deploying a BPM project to runtime, you must configure your Oracle Fusion applications to use the BPMN processes.

7.6.1 Configuring BPMN Processes in CRM Applications

After deploying a BPM project, you must use CRM Application Composer to add an object workflow that conditionally responds to a record modification event for the business object. See Section 4.2, "Editing Objects" for more information on adding object workflows.

7.6.2 Configuring BPMN Processes in HCM Applications

After deploying your BPM project to Oracle BPM runtime, you must register the BPMN processes using the Register Workforce Process page in the Oracle Fusion Workforce Lifecycle Manager (WLM). When you register a process, you must provide the name of the process as it appears in the Start menu of the Manage Workforce Process page. Depending on how your process is designed, you may also need to specify the parameters that are passed to the workforce process on startup.

For more information, see product-specific documentation in Oracle Fusion Applications Help.

8

Customizing Reports and Analytics

This chapter describes how to use Oracle Business Intelligence Suite (Oracle BI Suite) to customize and extend reports and analytics for Oracle Fusion Applications, including customizing Oracle BI Publisher layouts and data models, customizing Oracle BI Enterprise Edition analyses and dashboards, and extending the Oracle BI repository.

This chapter includes the following sections:

8.1 About Customizing Reports and Analytics

In Oracle Fusion Applications, reports and analytics are built using Oracle Business Intelligence:

- Reports are built with Oracle Business Intelligence Publisher (BI Publisher) and are usually highly-formatted, detailed documents.
- Analytics are analyses and dashboards built with Oracle Business Intelligence Presentation Services. Analyses are queries based on real-time, transactional or operational data that provide answers to business questions. Dashboards provide personalized views of corporate and external information. A dashboard consists of one or more pages that contain content, such as analyses, links to websites, BI Publisher reports, and so on.

8.2 Customizing Reports

This section describes how to use Oracle Business Intelligence Publisher to customize and extend reports for Oracle Fusion Applications. It includes the following sections:

- Section 8.2.7, "Securing Custom Reports and Related Components"
- Section 8.2.8, "Making Reports Available to Users in the Reports and Analytics Pane"
- Section 8.2.9, "Enabling Reports for Scheduling from the Reports and Analytics Pane"

8.2.1 About Customizing Reports

Reports extract the data from your applications and present it in the formats required for your enterprise. Reports provide the information you need for internal operations and statutory compliance; reports also provide the business documents for communicating with your customers. Many product-specific reports are provided with Oracle Fusion Applications; for example, the invoice register, the pick slip report, the payroll summary, the journals report, and the customer credit memo. To meet the specific needs of your enterprise, you may need to customize the reports provided or you may need to create new reports to capture and present different data.

Understanding the BI Publisher reporting architecture will help you to understand the report customization scenarios and tasks. A report in BI Publisher consists of the following components:

- A data model that defines the data source, data structure, and parameters for the report. A data model can be used by multiple reports. Each report has one data model.
- One or more layouts to define the presentation, formatting, and visualizations of the data. A report may have multiple layouts of the data model.
- A set of properties that specify runtime and formatting options

Optionally, a report may also include:

- Translations to provide localized versions of a report

8.2.1.1 About Tasks Required When Customizing Reports That Are Submitted by the Oracle Enterprise Scheduler

Many Oracle Fusion applications use the Oracle Enterprise Scheduler to submit report requests to BI Publisher. For reports that require parameter value input from users, the Oracle Enterprise Scheduler uses a parameter view object to collect and validate parameter values to send to BI Publisher.

After you customize the report in BI Publisher, additional tasks are required to set up the Oracle Enterprise Scheduler job definition to run your report from Oracle Fusion applications. If your report customization includes customizing parameters that are passed to BI Publisher, you will likely need to customize the parameter view object.

This chapter highlights the additional tasks required for these related components. You must also reference the following documents for full descriptions of how to customize them and how to configure the Oracle Enterprise Scheduler job definition to integrate the parameter view object:

- For customizing Oracle Enterprise Scheduler job definitions, see Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs."
- For information on how to customize view objects in Oracle Fusion Applications, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

- For the full description of view objects and creating them in JDeveloper, see the "Defining SQL Queries Using View Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

8.2.1.2 What You Can Customize

In many cases, BI Publisher reports delivered with Oracle Fusion applications will contain the appropriate data elements you expect, but may not provide the presentation of the data just as you would like it. BI Publisher enables you to customize the layouts for reports leveraging the prebuilt data models. If the reports provided by Oracle Fusion applications do not include the data you require, you can create a new report based on a custom data model.

> **Important:** Do not edit the predefined report objects. If you change a report and a subsequent patch includes a new version of the report, the patch overwrites any customizations. If subsequent patches do not include a new version of the report, the customizations are retained. When customizing reports, always make a copy of the original object and edit the copy.

Some common report customization scenarios are shown in the following table.

Customization Use Case	Described in
Edit the layout of a report provided with an application	"Customizing Layouts" on page 8-6
For example: Add your company logo to the Receivables Credit Memo report	
Add a new layout to a report provided with an application	"Customizing Layouts" on page 8-6
For example: Design a new form letter users can select when they run the Receivables Credit Memo	
Create a new report based on a data model provided with an application	"Creating Custom Reports" on page 8-24
Edit a data model provided with an application	"Editing Existing Data Models" on page 8-22
For example: Add a field to a data model	
Create a new data model	"Creating a New Data Model" on page 8-23
For example: Define a new query against Oracle Fusion applications tables	
Create a new report based on a custom data model	"Creating Custom Reports" on page 8-24

8.2.1.3 Related Report Customization Tasks

Depending on how a report is implemented in Oracle Fusion Applications and the type of customization you make you may also have to perform additional tasks to implement your custom report in the system.

If you create a new report and you wish to schedule this report through Oracle Enterprise Scheduler Service, you must create an Oracle Enterprise Scheduler Service job for the report. If you require Oracle Enterprise Scheduler Service to send parameter

values to the BI Publisher report via a parameter view object, you must also create the view object.

If you create a custom layout and you require translations of the layout, you must also provide the translations. Oracle Business Intelligence Publisher provides a tool for extracting the translation file for some layout types. The translation file can be translated into the required languages then uploaded to the report.

Table 8–1 provides links to related report customization tasks.

Table 8–1 Related Report Customization Tasks

Related Report Customization Task	Described in
Create the Parameter View Object for Oracle Enterprise Scheduler Service	"Customizing Parameters for Reports Submitted Through Oracle Enterprise Scheduler" on page 8-26
Create the Oracle Enterprise Scheduler Service job definition and job	"Tasks Required to Run Custom Reports with Oracle Enterprise Scheduler Service" on page 8-25
Provide translations	"Adding Translations" on page 8-25
Secure reports and related components	"Securing Custom Reports and Related Components" on page 8-27

8.2.1.4 Tools for Customizing Reports

Customize reports either within the Oracle BI Publisher application or using one of the tools or applications listed in Table 8–2. For the list of certified versions of third-party applications, see the "System Requirements and Certification" section in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

Table 8–2 Tools for Customizing Report Components

Component	Tool for Customizing
Report Data Model	BI Publisher's data model editor
Report properties	BI Publisher's report editor
Layout	See Table 8–3, " Tools for Customizing Layouts"

Table 8–3 shows the tools required to customize each template type.

Table 8–3 Tools for Customizing Layouts

Layout Template Type	Tool for Customizing
RTF template	Microsoft Word with BI Publisher's Template Builder for Word
BI Publisher template (XPT)	BI Publisher's layout editor
PDF template	Adobe Acrobat Professional
Excel template	Microsoft Excel with BI Publisher's Template Builder for Excel
eText Template	Microsoft Word

8.2.1.5 Before You Begin Customizing Reports

Before you customize reports, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1,

"Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with runtime customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

In addition, be familiar with the following BI Publisher-specific requirements:

Ensure that you have proper permissions for editing and creating Oracle Business Intelligence Publisher objects

To create or edit reports and report layouts requires the BI Author Role (or a role that includes the BI Author Role) as well as write permissions on the objects in the catalog to be edited.

To create or edit data models requires a custom role that includes the BI Author Role and the developDataModel permission (oracle.bi.publisher.developDataModel). Note that the ability to create BI Publisher data models allows the user to write and execute SQL, therefore implementers must consider carefully to whom they grant the developDataModel permission, and on which environments.

To create the custom role for editing data models, follow the guidelines in the "Configuring Roles" section in the *Oracle Fusion Applications Administrator's Guide*.

For more information about setting permissions in the catalog, see the "Managing Objects in the Oracle BI Presentation Catalog" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*.

Understand how the patching process for catalog objects impacts customizations

If a patch includes an update to a catalog object that was delivered with an Oracle Fusion application (for example, the Payables Invoice Register report) the patch will overwrite any customizations applied to the original report. To avoid overwriting a customization, do not customize a predefined Oracle Fusion application object in place; create a copy of the object and customize the copy.

Understand how permissions are set for and inherited by catalog objects

For a user to view a report, the user's role must have read permissions on every object referenced by a report. Permissions can be inherited from the folder in which the object resides.

For ease of maintenance, Oracle recommends that you place customized reports within the same folder as the original; or, if creating a new report that you place it within the same folder as other reports for the same job role.

If you choose to create new folders, bear in mind the catalog security permissions required (see the "Managing Objects in the Oracle BI Presentation Catalog" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*).

Be aware of any application-specific guidelines for customizing reports

See application-specific documentation in the Oracle Enterprise Repository for Oracle Fusion Applications.

Be aware of property settings that determine how the report can be run and viewed

Some reports are configured to run only through an external application or through the Oracle Enterprise Scheduler. While customizing a report, you may want to

configure it temporarily for viewing online to facilitate testing. See Task: Review Report Settings for Online Viewing for information about these settings.

Know how to navigate to Oracle Business Intelligence Presentation catalog objects

Navigate to the Oracle BI Presentation catalog as follows:

On the **Navigator**, under **Tools**, click **Reports and Analytics**. In the **Reports and Analytics** pane, click **Browse Catalog**.

Alternatively, log in to Oracle Business Intelligence directly (example: http://host:port/analytics/saw.dll).

Oracle Fusion Applications reporting objects are organized by product family in the catalog typically as follows:

- Top Level: Shared Folders
  - Product Family Folder (example: Financials)
    * Product folder (example: Payables)

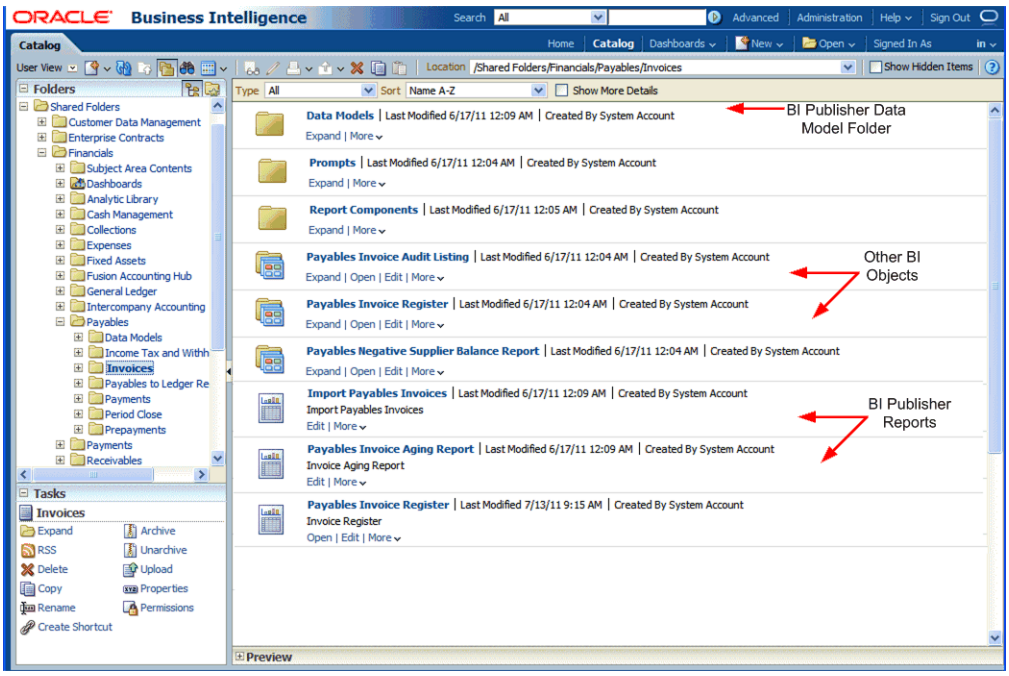
      --Report group folders (example: Invoices)

      -- Reports

      -- Data Model folder

Figure 8–1 shows the BI Presentation catalog.

Figure 8–1 BI Presentation Catalog



8.2.2 Customizing Layouts

The layout defines the presentation of the report data. All reports delivered with Oracle Fusion Applications include at least one predefined layout template file that defines the presentation components (such as tables and labeled fields) and maps the

elements from the data model to these components. The layout also defines font sizes, styles, borders, shading, and can also include images, such as a company logo.

To customize a layout, you edit the layout template. BI Publisher supports several types of templates to support different report layout requirements. Most of the templates delivered with Oracle Fusion Applications are rich text format (RTF) templates created using Microsoft Word. Some predefined templates are BI Publisher layout templates created using BI Publisher's layout editor. These are for interactive and more visually appealing layouts. A third type is the eText template, which is used specifically for electronic data interchange (EDI) and electronic funds transfer (EFT).

BI Publisher templates can also be created using Adobe PDF, Microsoft Excel, Adobe Flash, and XSL-FO.

Some examples of layout customizations are:

- Style changes only, no changes to data mapping

  This is the simplest type of customization. Examples are removing the predefined logo from the report and inserting your own or simply modifying colors and font styles. For these changes you can download the predefined template and edit it. Because there are no changes to the data mapping, style changes do not require sample data from your report; however, having sample data available will enable testing of the template from your desktop.

- Changes to mapped data elements within the existing layout

  An example of this type of customization is adding or removing a table column or data field from the report layout. For these changes you must have sample data to load to the layout editing tool. You can download the predefined template, load your sample data, insert the required elements, preview your template, then upload your customized template back to the report definition.

- New presentation of the data

  To create a new layout, start by opening the layout editing tool and loading the sample data to begin designing your custom layout.

Task: Generate Sample Data from the Report

The layout tools require sample data to enable the mapping of data fields to layout components in the report. You can generate sample data in the following ways:

- From the report data model
- From the report viewer

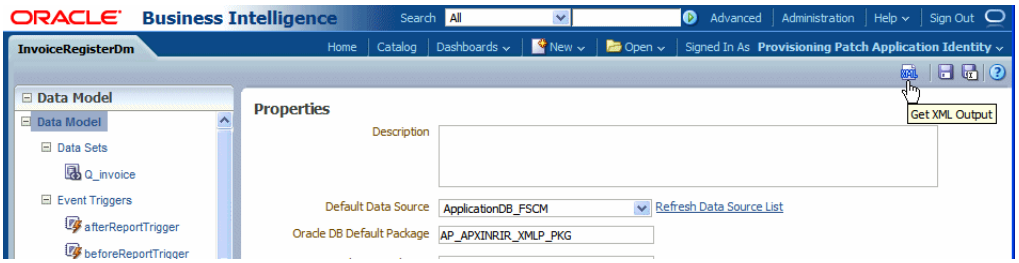
To generate sample data from the data model:

1. Navigate to the report data model in the BI Presentation catalog and click **Edit**.

   Tip: If you are not sure which data model is the source for a particular report, click the report **Edit** link. The data model is displayed in the upper left corner of the report editor.

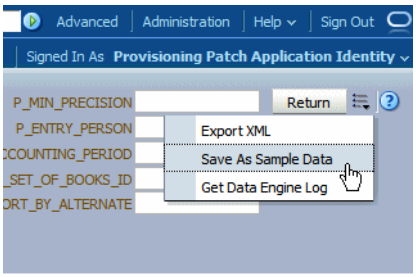
2. In the data model editor, click **Get XML Output** as shown in Figure 8–2.

Figure 8–2 Getting XML Output from a Data Model



3. Enter values for any required parameters, select the **Number of rows to return**, and click **Run**.
4. To save the sample data to the data model, click the **Actions** menu and then click **Save As Sample Data**, as shown in Figure 8–3.

Figure 8–3 Saving Sample Data to a Data Model



If you are designing an RTF template, you may also wish to **Export the XML** to save the file to a local client for use with the Template Builder for Microsoft Word.

5. Click **Return**. Then save the data model.

To save sample data from the report viewer:

1. Navigate to the report in the Business Intelligence catalog.
2. Click **Open** to run the report in the report viewer with the default parameters.
3. On the **Actions** menu, click **Export**, then click **Data**.
4. Save the data file.

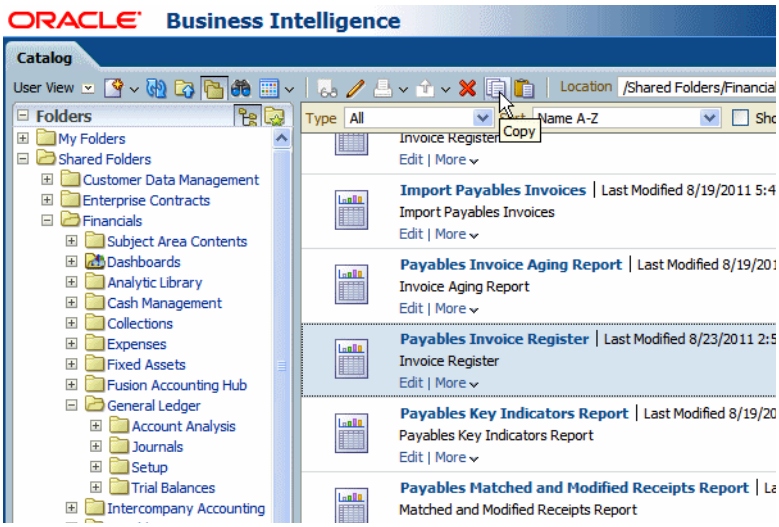
Task: Make a Copy of the Original Report

Navigate to the report in the BI Presentation catalog.

To make a copy of the report:

1. Select the report by clicking anywhere in the row. On the catalog toolbar, click **Copy** as shown in Figure 8–4.

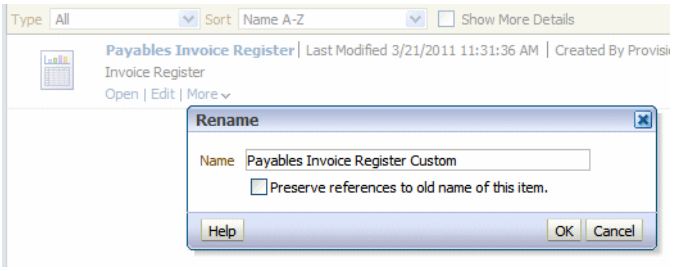
Figure 8–4 Copying a Predefined Report in the BI Presentation Catalog



2. Click **Paste** to place the copy in the current folder. When you paste into the same folder, the copy is created as "copy of *original report*".

3. Rename the copied report. For example: Payables Invoice Register Custom, as shown in Figure 8–5.

Figure 8–5 Renaming the Copied Report



4. (Optional) Update the report description. In the catalog, click the **Edit** link. In the report editor, click the **Properties** link at the top of the page. Enter the **Description** for your report, for example: "Payables invoice register report with custom layout".

Task: Review Report Settings for Online Viewing

Some reports are configured to view only through an external application or through the Oracle Enterprise Scheduler Service. If you wish to view your report online while you are customizing it, ensure that the following properties are set as follows.

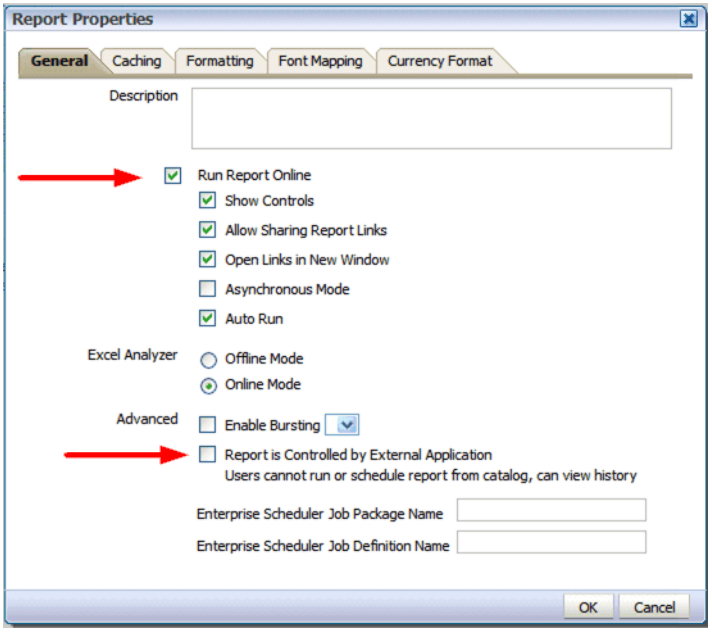
- Report Properties Settings
  - **Run Report Online** - must be enabled
  - **Report is Controlled by External Application** - must be disabled

  To navigate to the report **Properties** dialog:

  1. Navigate to your report copy in the catalog and click **Edit**.
  2. In the report editor, click the **Properties** link at the top of the page.

3. In the **Properties** dialog, select **Run Report Online** and clear **Report is Controlled by External Application**. These properties are shown in Figure 8–6.

Figure 8–6 Report Properties Dialog



- Layout Setting

  The layout setting **View Online** must be enabled.

  To view the layout settings:

  In the report editor, click **View a List**. Ensure that the **View Online** property is enabled.

Task: Edit or Create the Layout

To design an RTF layout, see Section 8.2.2.1, "Customizing RTF Templates."

To design a layout using the BI Publisher layout editor, see Section 8.2.2.2, "Customizing BI Publisher Templates."

To design one of the other supported layout types, see the corresponding chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*:

- Creating PDF Templates
- Creating Excel Templates
- Creating eText Templates

Task: Upload the Template File to the Report Definition

If you created a layout using the layout editor, the layout is automatically saved to the report definition and you can skip this step. For all other layout types upload the template file to the report definition.

To upload the template file to the report definition:

1. Navigate to your custom report in the catalog and click **Edit**.

2. On the report definition page, click **View a List**. On the table that lists the layouts, click **Create** (the "+" button).
3. Click **Upload** to upload the template file from your local directory.
4. Save the report definition.

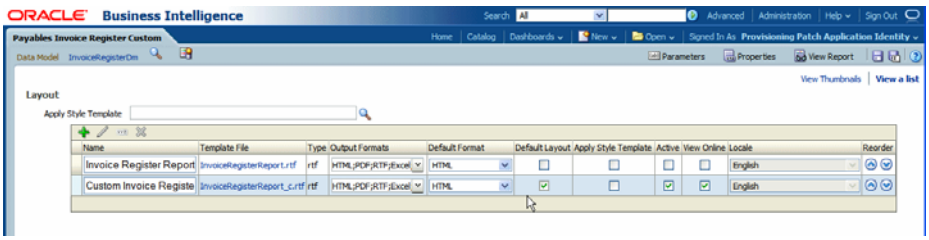
Task: Configure the Layout Settings

> **Tip:** To hide the original layout from users, clear the **Active** box.

To edit the layout settings:

1. In the report editor, click **View a List**. The List View is shown in Figure 8–7.

Figure 8–7 Report Layouts Shown in the List View



Set the following properties for your custom layout:

- Output Formats

  Output formats are the file formats available for the generated report, such as PDF, HTML, RTF, Excel. Depending on the requirements of a report you may want to limit the output formats available to users. The output formats available will depend on the Template File Type.

- Default Output Format

  If multiple output formats are available for the report, the default output format will be generated by default when the report is run in the report viewer.

- Default Layout

  If multiple layouts are available for the report, the default layout will be presented first in the report viewer. One and only one layout must be selected as the default layout.

- Apply Style Template

  If a style template is assigned to this report, use this field to apply the style template to the layout.

- Active

  Active layouts are displayed to report consumers. Inactivate the layout to make it unavailable to report consumers.

- View Online

  Layouts that can be viewed online are available to report consumers from the report viewer. If this property is not checked, the layout is available only for scheduled jobs.

Task: Delete a Layout

To delete a layout from the report:

1. In the report editor, click **View a List**.
2. Locate the layout that you wish to delete in the table and click anywhere within its row to select it.
3. Click the **Delete** toolbar button. Click **OK** in the confirmation dialog.

Task: (Conditional) Create the Oracle Enterprise Scheduler Service Job to Run the Custom Report

If this report replaces a report that relies on an Oracle Enterprise Scheduler Service job for submission, you must create a custom Oracle Enterprise Scheduler Service job to point to your report copy. See Section 8.2.6, "Tasks Required to Run Custom Reports with Oracle Enterprise Scheduler Service."

Task: (Conditional) Enable Access to the Report Through the Reports and Analytics Pane

See Section 8.2.8, "Making Reports Available to Users in the Reports and Analytics Pane."

8.2.2.1 Customizing RTF Templates

Most templates delivered with Oracle Fusion Applications are RTF templates. An RTF template is a rich text format file that contains the layout instructions for BI Publisher to use when generating the report. RTF templates are created using Microsoft Word. BI Publisher provides an add-in to Microsoft Word to facilitate the coding of layout instructions. For more information see the "Creating RTF Templates Using the Template Builder for Word" chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

Note: If you are designing a new layout for the report, consider using the BI Publisher layout editor. The layout editor is an online layout editing tool launched from the report editor. See Section 8.2.2.2, "Customizing BI Publisher Templates."

Before you begin:

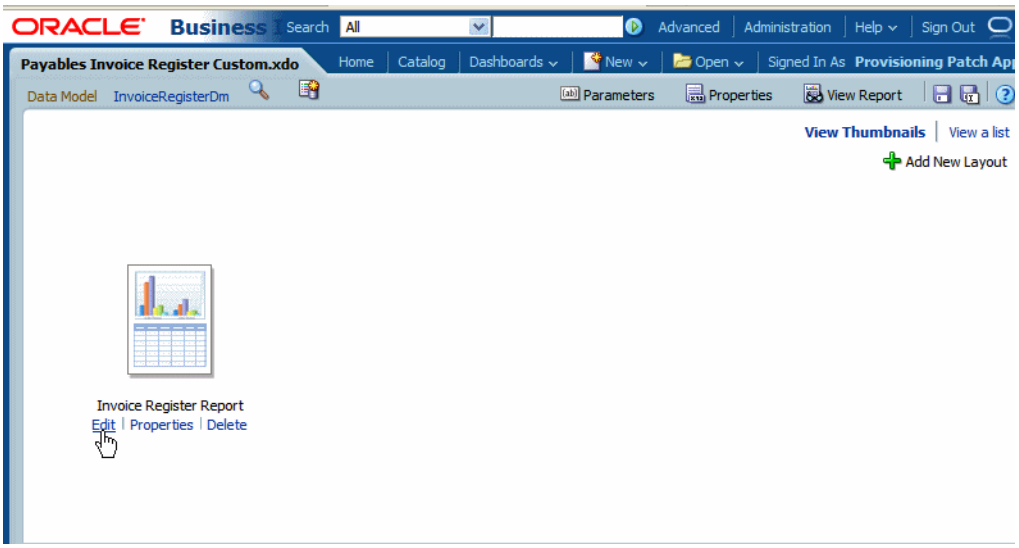
Ensure that your local client has a supported version of Microsoft Word. BI Publisher provides the Template Builder for Microsoft Word to facilitate RTF template design. Download the tool from the Oracle Business Intelligence home page. For more information see the "Creating RTF Templates Using the Template Builder for Word" chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

Task: Download the Template File

If you are creating a new layout, skip this step.

If you are modifying a predefined layout, navigate to the report in the catalog and click **Edit**. In the report editor, click the **Edit** link of the layout to download the RTF file to your local client, as shown in Figure 8–8.

Figure 8–8 Downloading the Predefined Layout



Task: Edit the RTF File in Microsoft Word

Open the downloaded RTF template file in Microsoft Word; or, if you are creating a new template, open Microsoft Word.

Load the sample data to the Template Builder for Word Add-in.

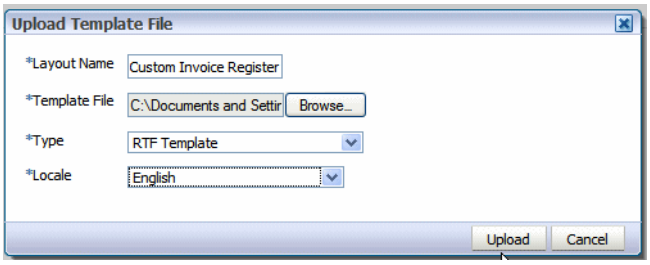
Edit or create the layout following the guidelines in the Template Builder help or in the "Creating RTF Templates Using the Template Builder for Word" chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

Ensure that you save the file as Rich Text Format (rtf).

Task: Upload the Template File to the Report Definition

In the catalog, open the report in the report editor and click **View a List**. On the table that lists the layouts, click **Create** (the "+" button). Click **Upload** to upload the RTF file from your local directory as shown in Figure 8–9.

Figure 8–9 Uploading the Template File



Task: Add Translations for the Layout

If this report requires translations, see Section 8.2.5, "Adding Translations."

8.2.2.1.1 Customizing an RTF Template: Examples This section includes two examples of RTF template customizations:

- Changing the Inserted Logo in a Predefined Template

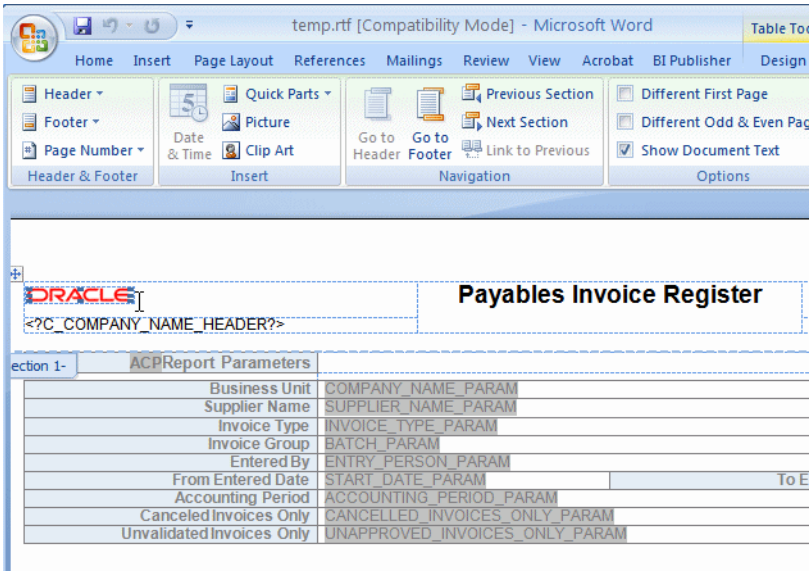
- Customizing an RTF Template Using an Existing Data Model

Example 8–1 Changing the Inserted Logo in a Predefined Template

The Payables Invoice Register report layout includes a standard logo in the report header. To change the inserted logo using Microsoft Word 2007:

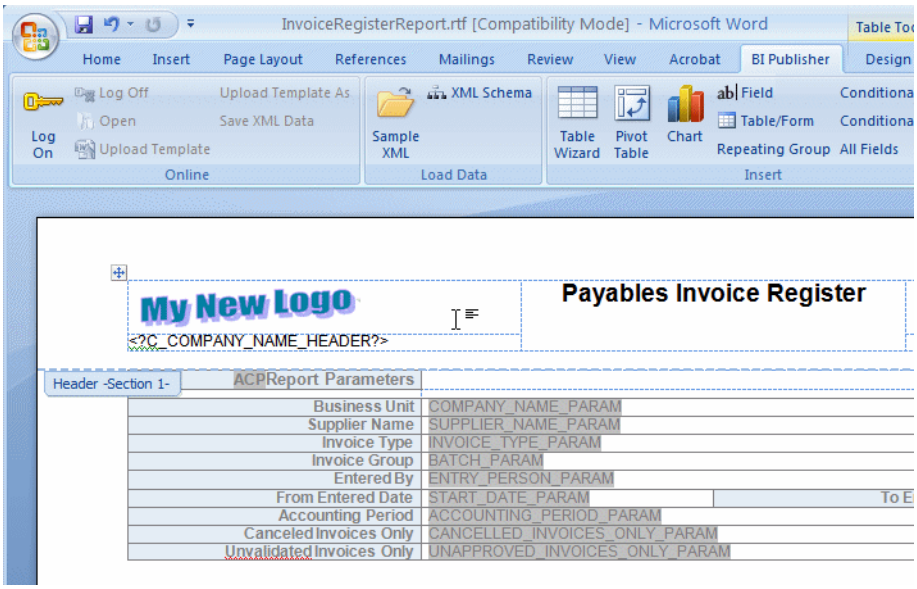
1. In Microsoft Word, delete the Oracle logo as shown in Figure 8–10.

Figure 8–10 Selecting the Oracle Logo Prior to Deletion



2. On the **Insert** tab in the **Illustrations** group click **Picture**. Select your company logo image file to insert it to the Word document. Resize the image as necessary. An example is shown in Figure 8–11.

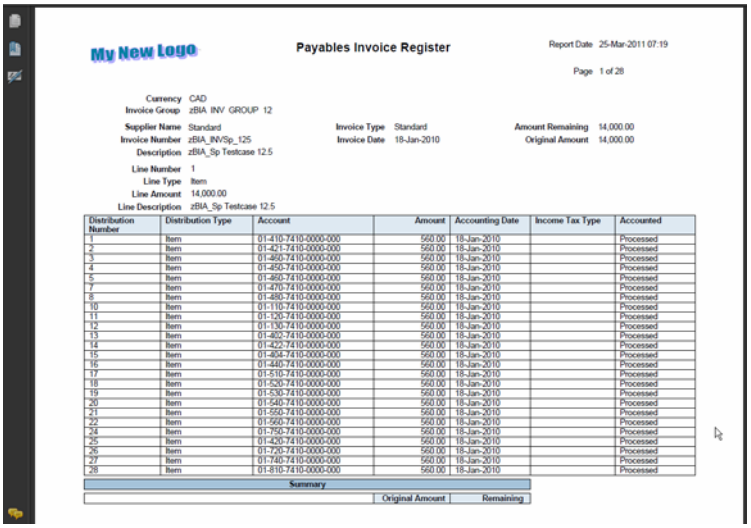
Figure 8–11 Inserting the Logo in Microsoft Word



Tip: If the template file includes section breaks, you must insert the new logo for each section header.

3. If you downloaded sample data, you can test the template on your desktop: On the BI Publisher tab, in the **Preview** group click **PDF**. The Template Builder will apply the sample data you loaded and generate a PDF output document, as shown in Figure 8–12.

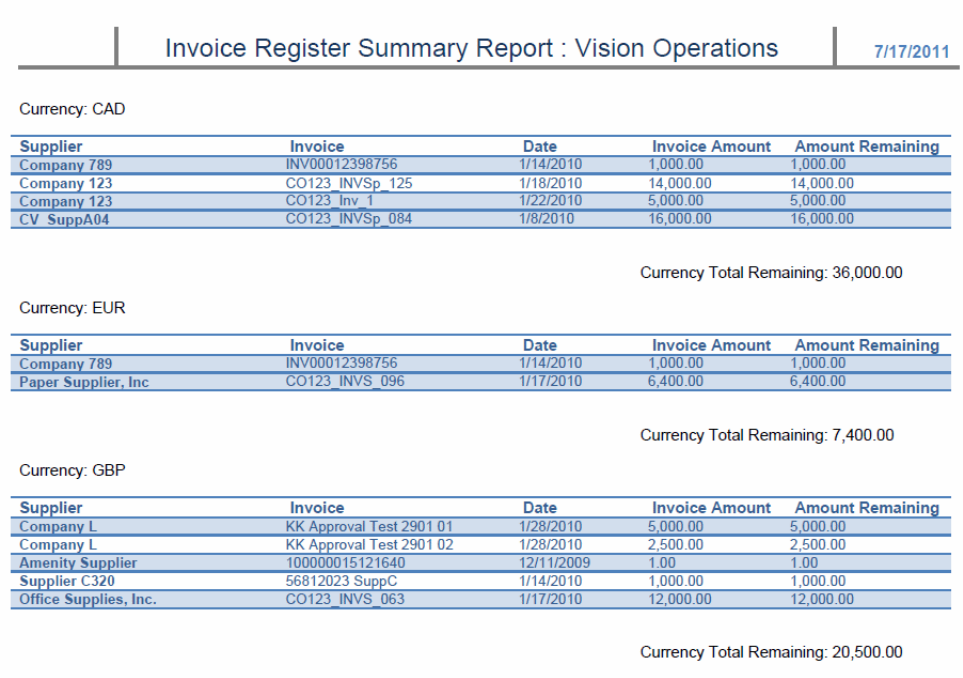
Figure 8–12 Preview of Custom Layout Template



Example 8–2 Customizing an RTF Template Using an Existing Data Model

This example demonstrates the general steps for customizing an RTF template using an existing data model. In this example, the Payables Invoice Register data model is used to create a new layout to display a summary for each currency. This example demonstrates general report layout concepts and features of the BI Publisher Template Builder for Microsoft Word. Follow the steps in this topic and the guidelines in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

The final report layout is shown in the following figure:



Invoice Register Summary Report : Vision Operations 7/17/2011

Currency: CAD

Supplier	Invoice	Date	Invoice Amount	Amount Remaining
Company 789	INV00012398756	1/14/2010	1,000.00	1,000.00
Company 123	CO123_INVSp_125	1/18/2010	14,000.00	14,000.00
Company 123	CO123_Inv_1	1/22/2010	5,000.00	5,000.00
CV_SuppA04	CO123_INVSp_084	1/8/2010	16,000.00	16,000.00

Currency Total Remaining: 36,000.00

Currency: EUR

Supplier	Invoice	Date	Invoice Amount	Amount Remaining
Company 789	INV00012398756	1/14/2010	1,000.00	1,000.00
Paper Supplier, Inc	CO123_INVS_096	1/17/2010	6,400.00	6,400.00

Currency Total Remaining: 7,400.00

Currency: GBP

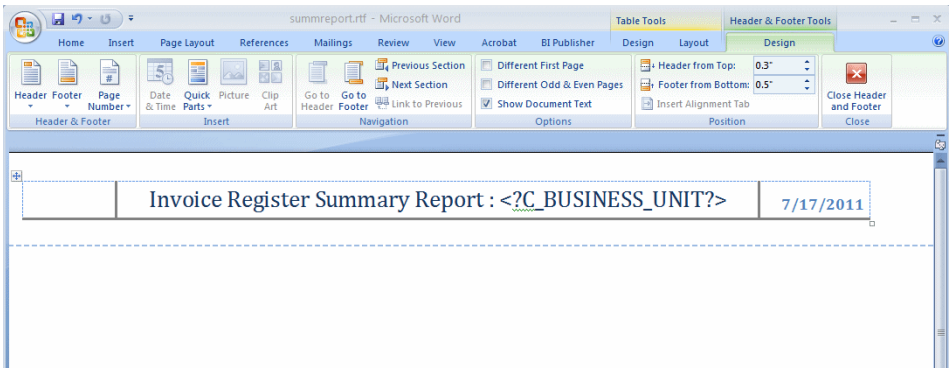
Supplier	Invoice	Date	Invoice Amount	Amount Remaining
Company L	KK Approval Test 2901 01	1/28/2010	5,000.00	5,000.00
Company L	KK Approval Test 2901 02	1/28/2010	2,500.00	2,500.00
Amenity Supplier	100000015121640	12/11/2009	1.00	1.00
Supplier C320	56812023 SuppC	1/14/2010	1,000.00	1,000.00
Office Supplies, Inc.	CO123_INVS_063	1/17/2010	12,000.00	12,000.00

Currency Total Remaining: 20,500.00

To create this layout:

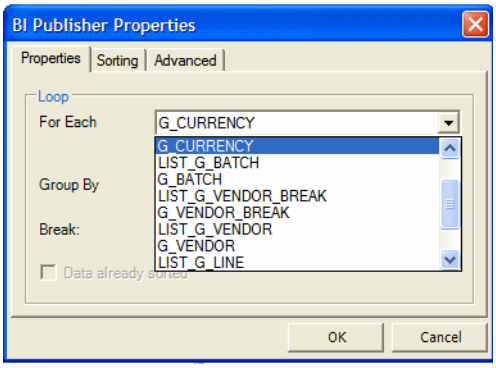
1. Download to your local client sample data from the Payables Invoice Register data model (InvoiceRegisterDM).
2. In Microsoft Word, on the **BI Publisher** tab, in the **Load Data** group, click **Sample Data**. Select your downloaded file and click OK.
3. Using Microsoft Word functionality, insert the page header. Type the text for the header, and insert the field for the Business Unit as shown in Figure 8–13.

Figure 8–13 Inserting the Header to the RTF Template



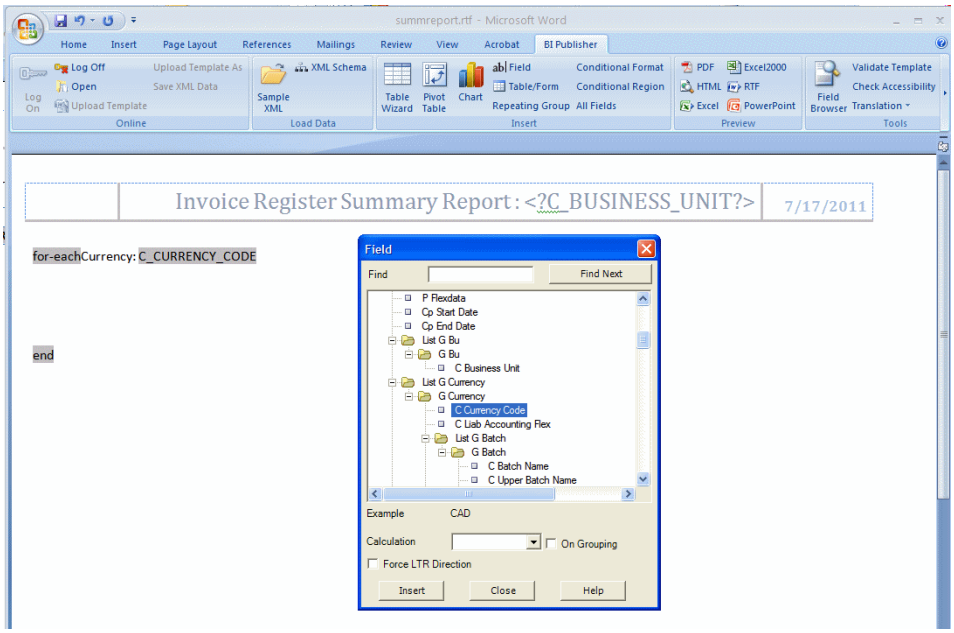
4. In this example, the report will repeat the table of invoices for each occurrence of the currency code. To create this behavior, insert a repeating group based on the element C_CURRENCY_CODE. To insert the repeating group:
   a. On the **BI Publisher** tab, in the **Insert** group, click **Repeating Group**.
   b. In the **BI Publisher Properties** dialog, select the G_CURRENCY group as shown in Figure 8–14.

Figure 8–14 Selecting a Repeating Group



c. To display the field value, type the text "Currency:" after the for-each tag. To insert the element from the data: On the **BI Publisher** tab, in the **Insert** group, click **Field**. The **Field** dialog will display. Select the C_CURRENCY_CODE element, as shown in Figure 8–15.

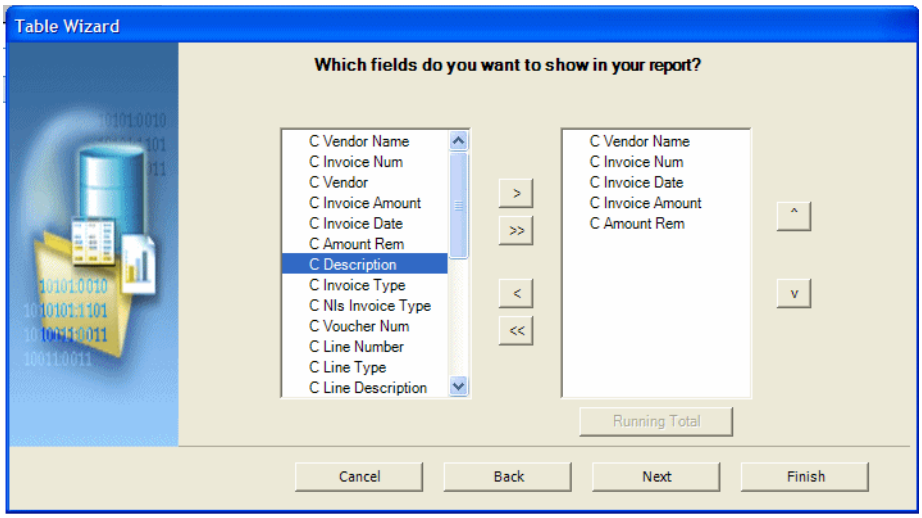
Figure 8–15 Inserting the Currency Code Field



5. Use the table wizard to insert the table: On the **BI Publisher** tab in the **Insert** group, click **Table Wizard**. Make the following selections:

- For the report format, select **Table**.
- For the **Data Set** select the VENDOR group.
- Select the fields to display in your table: C Vendor Name, C Invoice Num, C Invoice Date, C Invoice Amount, and C Amount Rem, as shown in Figure 8–16.

Figure 8–16 Selecting Fields for the Table



- Click **Finish**. The inserted table will display with the column names from the data; also, if you preview the report, you will notice that no formatting is applied to number and date fields.

6. Edit the column names by simply editing the text in the column header row.
7. Apply formatting to the date and number fields. To apply formatting to the date field:

Note: This example shows simple number formatting. If your report requires locale-driven number and date formatting, see the topic "Formatting Numbers, Dates, and Currencies" in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

---

a. Right-click the date field in the table and select **BI Publisher** then **Properties** from the menu.

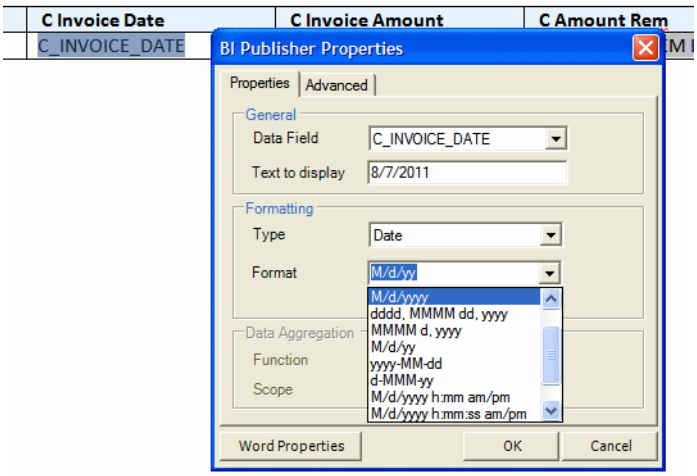
b. In the **BI Publisher Properties** dialog, update the following (shown in Figure 8–17):

   Set the **Type** to Date.

   Select the date **Format** from the list.

   Enter **Text to display**, for example: 8/7/2011.

Figure 8–17 Formatting the Date Field



To apply formatting to a number field:

a. Right-click the amount field in the table and select **BI Publisher** then **Properties** from the menu.

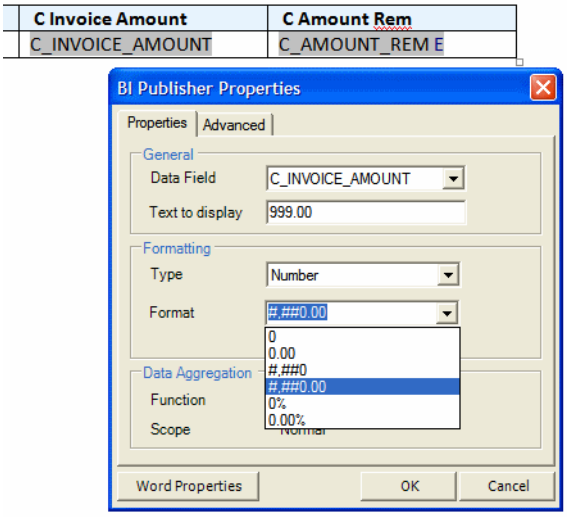
b. In the **BI Publisher Properties** dialog, update the following (shown in Figure 8–18):

Set the **Type** to Number

Select the number **Format** from the list.

Enter **Text to display**, for example: 999.00.

Figure 8–18 Formatting the Number Field



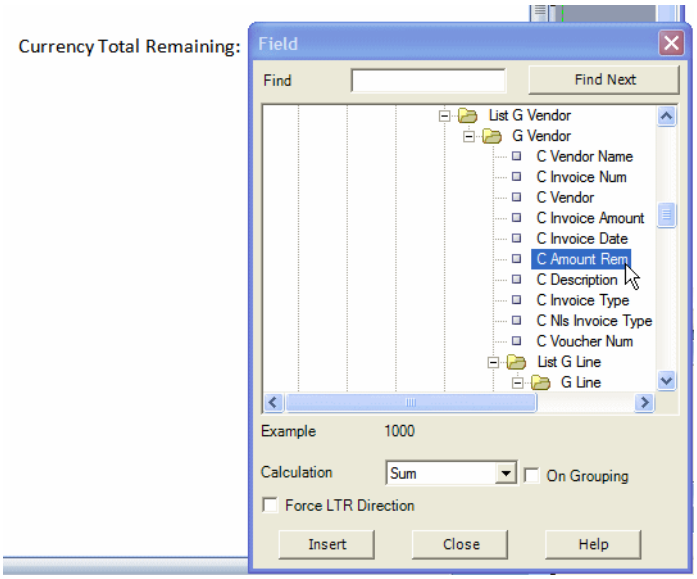
8. To display the total for each currency, enter the text: "Currency Total Remaining:" beneath the table, but inside the for-each / end tags. Insert the field as follows:

a. On the **BI Publisher** tab, in the **Insert** group, click **Field**.

b. In the **Insert Field** dialog, click the "C Amount Rem" field (under the G Vendor group).

c. In the **Calculation** field, select **Sum** from the list as shown in Figure 8–19.

Figure 8–19 Inserting and Summing the Total Field



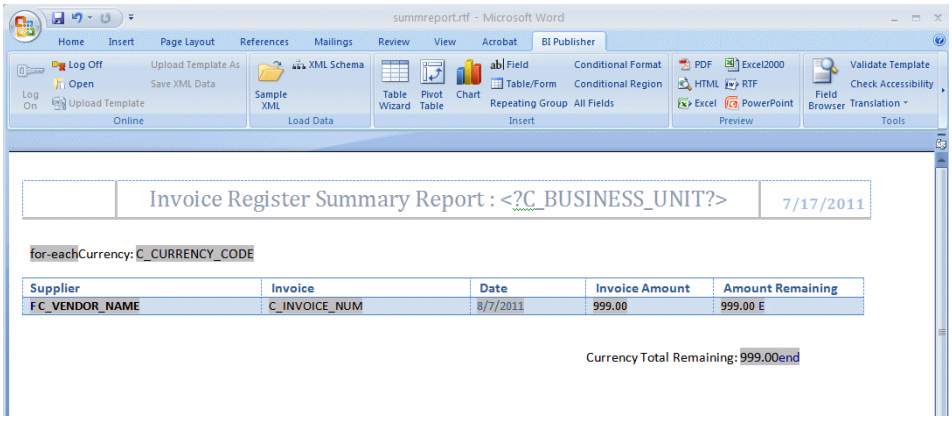
d. Click **Insert**.

e. Format the field as a number, as described in Step 7.

9. Perform any additional formatting of colors, fonts, borders, and text strings using Microsoft Word functionality.

The completed template will appear as shown in Figure 8–20. To preview the template, click the **PDF** button in the BI Publisher **Preview** group.

Figure 8–20 Finished RTF Template



8.2.2.2 Customizing BI Publisher Templates

BI Publisher templates are created using the BI Publisher Layout Editor - a design tool that provides an intuitive, drag and drop interface for creating pixel perfect reports in PDF, RTF, Excel, PowerPoint, and HTML. It also provides dynamic HTML output that supports lightweight interaction through a browser.

BI Publisher layouts are best suited for reports of simple to medium complexity. The interactive view is only available for BI Publisher layouts, therefore choose this layout type when you want your report consumers to interact with the report (change sorting, apply filters, and so on).

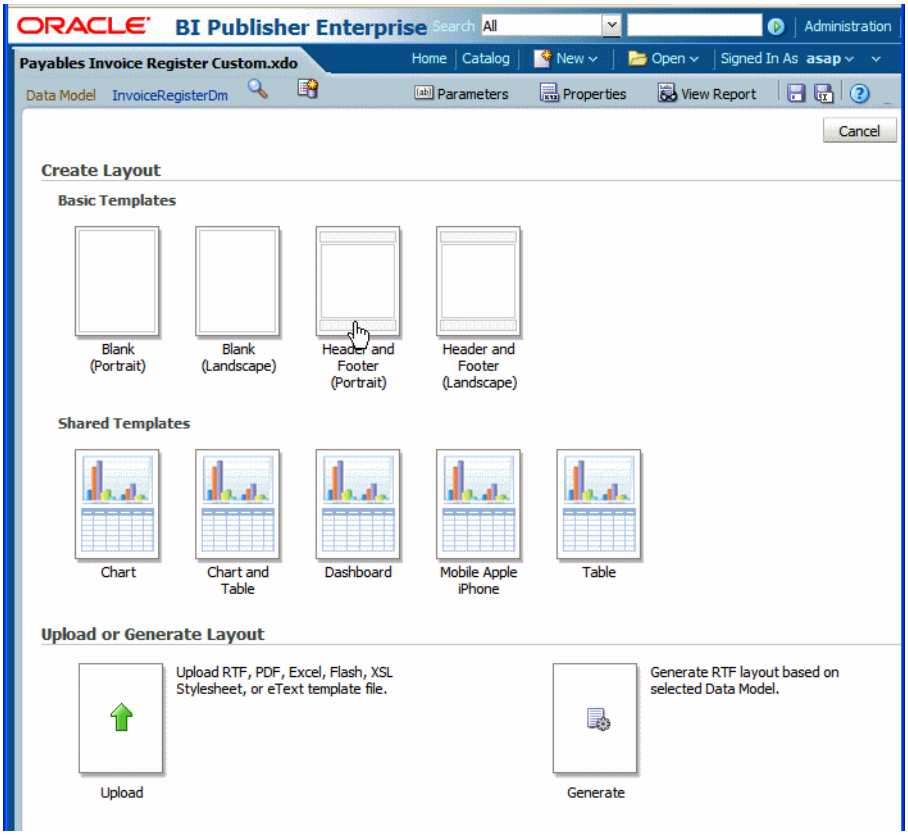
Before you begin:

The BI Publisher layout editor requires the data model to include sample data. To save sample data to the data model, you must generate data according to the first option described in "Task: Generate Sample Data from the Report".

Task: Launch the Layout Editor from the Report Definition

Navigate to the report and click **Edit**. Click **Add New Layout**. Under the **Create Layout** region, click one of the Basic Templates or Shared Templates to launch the layout editor, as shown in Figure 8–21.

Figure 8–21 Selecting a Boilerplate to Launch the Layout Editor



Task: Create and Save the Layout

Create the layout using the guidelines in the online help or in the "Creating BI Publisher Layout Templates" chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*. Click **Save** to save the layout to the report definition.

Task: Add Translations for the Layout

If this report requires translations, see Section 8.2.5, "Adding Translations."

8.2.3 Customizing Data Models

A data model defines the source and structure of the data for a report. At runtime BI Publisher executes the data model to supply the XML data for a report. Create a custom data model when the data models supplied by your application do not provide the data required in your report. If you need to customize the data that is captured by the report data model, you can either edit an existing data model or create a custom data model.

Before You Begin: Understand the Use of Parameter View Objects with Oracle Enterprise Scheduler

If the report requires user input for parameter values and the report is submitted by the Oracle Enterprise Scheduler to BI Publisher, the Oracle Enterprise Scheduler uses a parameter view object to present and validate parameter values in the Oracle Fusion application. The values are then sent by Oracle Enterprise Scheduler to BI Publisher for execution of the data model.

In the parameter view object the parameters are defined as attributes and must be named incrementally as ATTRIBUTE1, ATTRIBUTE2, and so on. In the BI Publisher data model, you must define the parameters in the same order as they are defined in the parameter view object. The mapping of the value passed by Oracle Enterprise Scheduler to the BI Publisher data model is by order alone.

For example, in a BI Publisher data model assume you have defined the following parameters in this order:

- P_BUSINESS_UNIT
- P_VENDOR_ID
- P_INVOICE_TYPE

In the parameter view object you must define ATTRIBUTE1 to retrieve the value for P_BUSINESS_UNIT; ATTRIBUTE2 to retrieve the values for P_VENDOR_ID; and ATTRIBUTE3 to retrieve the values for P_INVOICE_TYPE.

Ensure that any edits you make to the ordering of parameters in the parameter view object, the BI Publisher data model, or the Oracle Enterprise Scheduler job definition are reflected in all places. See Section 8.2.6.2, "Customizing Parameters for Reports Submitted Through Oracle Enterprise Scheduler."

8.2.3.1 Editing Existing Data Models

Do not directly edit a data model delivered with an Oracle Fusion application. Instead, make a copy and edit the copy. To ensure that the proper privileges are inherited by the copied object, maintain the copy in the same folder as the original.

Task: Copy the Existing Data Model

Navigate to the data model in the BI Presentation catalog. To make a copy:

1. Click anywhere in the object's row to select the data model.
2. On the catalog toolbar, click the **Copy** toolbar button; then click the **Paste** toolbar button to paste the copy into the same folder.
3. To rename the copy, click the **More** link, and then click **Rename**. Enter the new name, for example: InvoiceRegisterDM Custom.

Task: Customize the Data Model in the Data Model Editor

Click the **Edit** link in the catalog to open the data model in the data model editor. See the "Using the Data Model Editor" chapter in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)* for general information about using the data model editor and the specific chapter for the component you wish to edit, for example:

- To add or delete a field from the SQL query, see the topic "Editing an Existing Data Set" in the chapter "Creating Data Sets."
- To add or delete a bursting definition, see the chapter "Adding Bursting Definitions."
- To edit parameters, see "Adding Parameters and Lists of Values."

> **Important:** If the report will use the Oracle Enterprise Scheduler to collect parameter values from end users, then the parameter logic, list of values and display attributes that are presented in the application interface are defined in a parameter view object and the Oracle Enterprise Scheduler job definition. The parameter values are then passed to BI Publisher when the job is submitted.
>
> In this case, edit the parameters in the parameter view object then ensure that the parameters in the BI Publisher data model are in the same order as they are defined in the parameter view object. Do not create the list of values in the BI Publisher data model for reports submitted via Oracle Enterprise Scheduler.

8.2.3.2 Creating a New Data Model

To create a data model:

1. Open the Data Model editor.

   On the global header, click **New**, then click **Data Model** to open the data model editor.

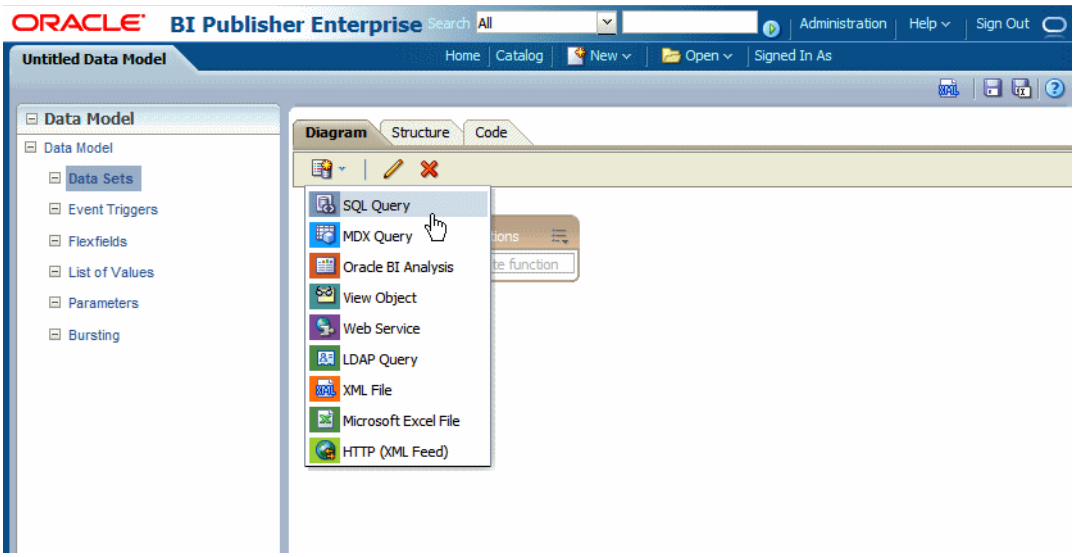
2. Configure the data model properties. For more information see the "Setting Data Model Properties" section in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)*.

3. If your data model will include event triggers you must enter the **Oracle DB Default Package**.

4. Create the data set.

   You will most likely create data sets from SQL queries against your Oracle Fusion application data tables. The data model editor also supports using an Oracle BI analysis as its source of data as well as entity view objects created in Oracle JDeveloper. See the "Creating Data Sets" chapter in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)* for more information about all supported data set types and how to create them. Figure 8–22 shows the menu of data set types available.

Figure 8–22 Creating a SQL Data Set



5. Create the optional components.

   A data model can include the following components:

   - Event triggers
   - Flexfields
   - Lists of values
   - Parameters

   Important: See "Before You Begin: Understand the Use of Parameter View Objects with Oracle Enterprise Scheduler" for information about how to define parameters in the BI Publisher data model when the report is submitted by the Oracle Enterprise Scheduler.

   - Bursting definition

   See the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)* for details on creating these components.

6. Add sample data to your data model. See "Task: Generate Sample Data from the Report" for the steps for adding sample data to a data model.

8.2.4 Creating Custom Reports

Create a custom report when the reports delivered with your Oracle Fusion application do not provide the data you need; or, if you want to use a predefined data model but change other properties of the report.

Task: Create the Data Model

If you are using a predefined data model, skip this step.

To create a data model, see Section 8.2.3.2, "Creating a New Data Model."

Task: Create the Report Definition in the Catalog and Choose the Data Model

On the global header, click **New**, then click **Report**. Select the data model to use for the data source for this report.

Task: Create and Add the Layout

Follow the tasks described in the section "Creating Custom Layouts" beginning with "Task: Edit or Create the Layout".

Task: Configure Report Properties

You can configure a variety of properties to set specific formatting, caching, and processing options for your report. To access the **Properties** dialog, click **Properties** in the report editor. For information on report properties see the "Configuring Report Properties" section in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

Task: (Conditional) Create an Oracle Enterprise Scheduler Job to Run the Custom Report

To schedule this report from an Oracle Fusion application, you must create an Oracle Enterprise Scheduler job and job definition. This may also require creating a parameter view object. See Section 8.2.6, "Tasks Required to Run Custom Reports with Oracle Enterprise Scheduler Service."

Task: (Conditional) Enable Access to the Report Through the Reports and Analytics Pane

See Section 8.2.8, "Making Reports Available to Users in the Reports and Analytics Pane."

8.2.5 Adding Translations

Template translation is a feature of BI Publisher that enables the extraction of translatable strings from a single RTF-based template or a single BI Publisher layout template. Use this option when you need multiple translations of the final report document; for example, you need to generate invoices for both German and French customers.

For information on adding translations for your custom report layouts, see the "Translating Individual Templates" chapter in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

8.2.6 Tasks Required to Run Custom Reports with Oracle Enterprise Scheduler Service

If you created a new report, to run it using Oracle Enterprise Scheduler, you must create a new Oracle Enterprise Scheduler job definition. If you customized an existing report (for example, added a custom layout) for which an Oracle Enterprise Scheduler job definition was defined, you will need to create a new job definition to point to the custom report.

Creating a custom Oracle Enterprise Scheduler job definition is described in Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs." Information specific to creating a custom BI Publisher job is in the section: "Section 14.2.2, "Extending a Custom Oracle BI Publisher Oracle Enterprise Scheduler Job."

This section summarizes the tasks:

- Creating a New Oracle Enterprise Scheduler Job Definition

- Customizing Parameters for Reports Submitted Through Oracle Enterprise Scheduler

8.2.6.1 Creating a New Oracle Enterprise Scheduler Job Definition

To create an Oracle Enterprise Scheduler Job Definition:

Follow the instructions for creating a job definition in Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs." Note the following for BI Publisher jobs:

- **Job Type**: Select BIPJobType
- **ReportID**: Enter the path to the report in the BI Presentation catalog, starting with the folder beneath Shared Folders; for example: Financials/Payables/Payables InvoiceRegisterCustom.xdo

> **Tip:** Ensure that you include the .xdo extension for the report object.

- **Default Output**: Select a default output format.
- **Bursting Job**: If the output from this job is to be burst, select this box. The BI Publisher report must have a bursting definition to use this option. When the report is executed, the output and delivery options are determined by the bursting definition. For information on setting up a bursting definition, see the "Adding Bursting Definitions" chapter in the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)*.
- Define the property `parametersVO` to point to the parameter view object you defined, if your custom job requires parameter input.

8.2.6.2 Customizing Parameters for Reports Submitted Through Oracle Enterprise Scheduler

The parameter view object is a view object used by Oracle Enterprise Scheduler to collect user input for report parameters that the Oracle Enterprise Scheduler then sends to BI Publisher. The parameter view object defines the display of the parameters in the Oracle Enterprise Scheduler interface and performs validation of the input. Use JDeveloper to edit the parameter view object.

Task: Create or Customize the Parameter View Object

To customize a parameter view object, see the following information:

- Chapter 11, "Customizing and Extending ADF Application Artifacts" for information about how to customize view objects in Oracle Fusion Applications.
- "Defining SQL Queries Using View Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for the full description view objects and creating them in JDeveloper.

Task: Align Parameter Order in the BI Publisher Data Model

Ensure that the attributes defined for the parameters in the view object are in the same order as the parameters in the BI Publisher data model. See "Before You Begin: Understand the Use of Parameter View Objects with Oracle Enterprise Scheduler" for more information.

Task: Integrate the Parameter View Object in the Oracle Enterprise Scheduler Job Definition

Follow the instructions for creating a job definition in Section 14.2, "Extending Custom Oracle Enterprise Scheduler Jobs Using Existing Oracle Fusion Applications.".

8.2.7 Securing Custom Reports and Related Components

When you create a custom report you may wish to create a custom role to enable only users that have been assigned the role to run the report. If you have also created an Oracle Enterprise Scheduler job to run the report, your users must also be assigned execution permissions for the job. All the tasks in this section are required when you create a custom role.

If you choose to enable the custom report for an existing role, ensure that this role is granted permissions on the objects in the BI Presentation catalog and on the Oracle Enterprise Scheduler job definition. In this case, you need only perform the tasks: Task: Configure Permissions in the BI Presentation Catalog and Task: Grant Permissions to the Oracle Enterprise Scheduler Job Definition.

Task: Create the Custom Report Role in the OBI Stripe

Only a system administrator can create a new role, and optionally include the role in an existing role hierarchy. For information about creating application roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Configure Permissions in the BI Presentation Catalog

Read permissions must be granted to every object in the BI Presentation catalog that is used in the report. This will always include at least two objects: the report and the data model. If your report also references a subtemplate or a style template, you must also grant read permissions on those objects. The report object requires additional grants to run, schedule, and view output.

If you create the custom report within an existing product folder, for example Payables/Invoices, the report will inherit permissions that are granted on all objects in the folder. You may wish to delete permissions from your custom report.

See "Managing Objects in the BI Presentation Catalog" in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information about catalog permissions.

Configuring permissions in the BI Presentation catalog consists of the following subtasks:

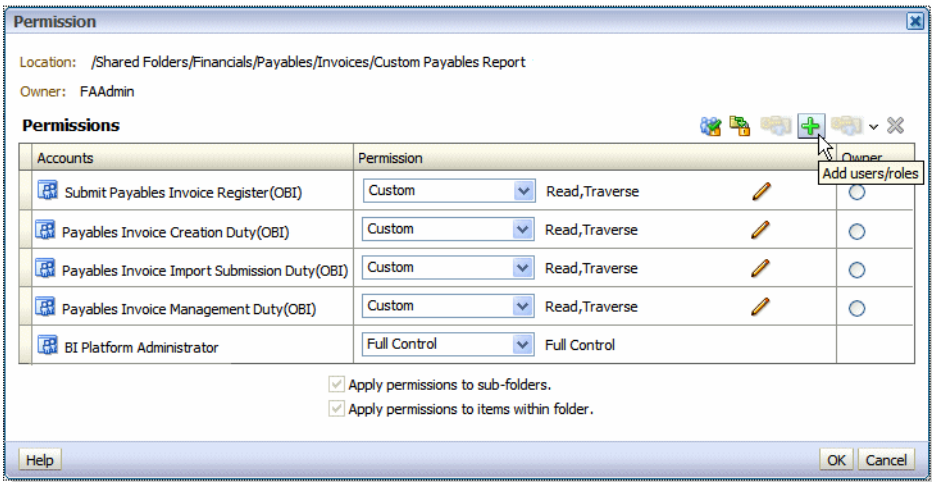
- "Grant the Role Permissions to the Report"
- "Grant the Role Permissions to the Data Model and Other Referenced Objects"
- "Delete Permissions"

Grant the Role Permissions to the Report

To grant permissions in the catalog:

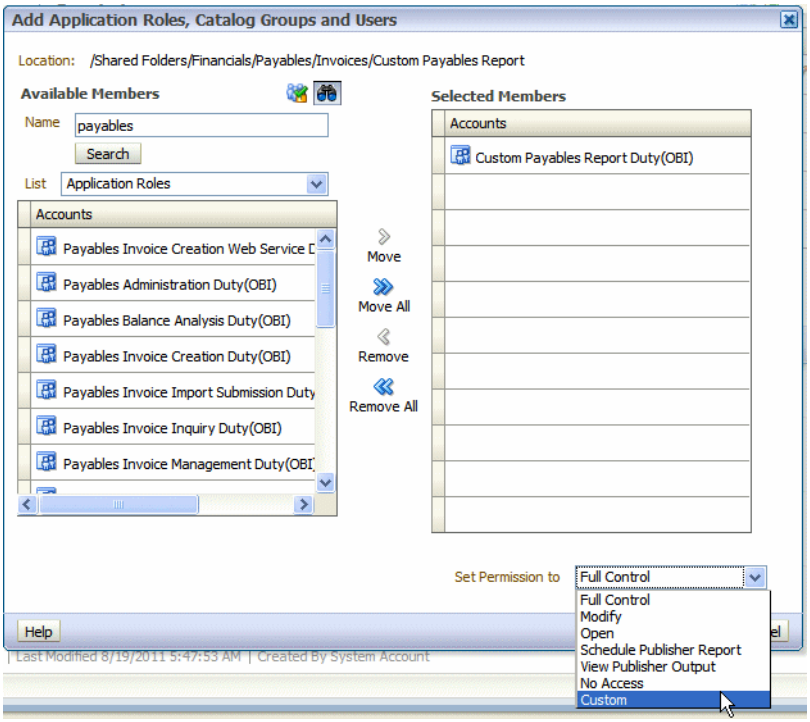
1. Navigate to the report in the catalog and click **More** and then click **Permissions**. The **Permissions** dialog launches and the inherited permissions are shown.
2. In the **Permissions** dialog, click **Add users/roles** as shown in Figure 8–23.

Figure 8–23 Detail of Add Users/ Roles in the Permission Dialog



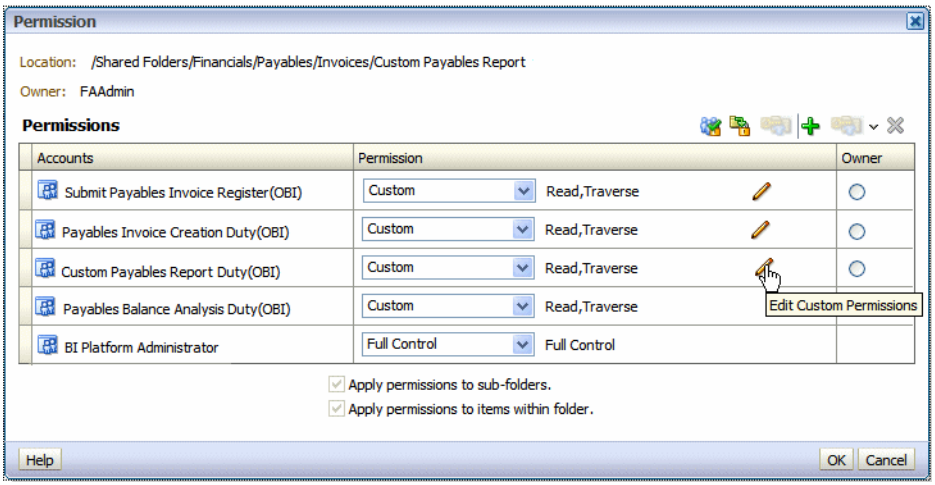
3. In the **Add Application Roles, Catalog Groups and Users** dialog, search for your custom role and use the shuttle buttons to move it to the **Selected Members** list. In the **Set Permission to** list, select **Custom**, as shown in Figure 8–24 and then click OK.

Figure 8–24 Adding a Role to the Selected Members List



4. In the **Permissions** dialog, locate the role you added and click **Edit** as shown in Figure 8–25.

Figure 8–25 Editing the Permissions Assigned to a Role

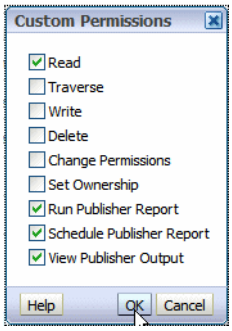


5. In the **Custom Permissions** dialog, select the permissions to enable. Typically, you will enable the following for a BI Publisher report:

   - Read - gives authority to access, but not modify, the report
   - Run Publisher Report - gives authority to read, traverse the folder that contains the report, and run the report.
   - Schedule Publisher Report - gives authority to read, traverse the folder that contains the report, and schedule the report
   - View Publisher Output - enables the user to view the report output generated by the scheduler

   The **Custom Permissions** dialog is shown in Figure 8–26.

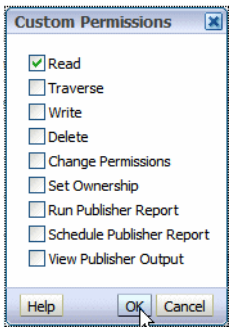
Figure 8–26 Setting Custom Permissions on the Report



Grant the Role Permissions to the Data Model and Other Referenced Objects

1. Navigate to the data model in the catalog and click **More** and then click **Permissions**. The **Permissions** dialog launches and the inherited permissions are shown.
2. Follow the instructions in "Grant the Role Permissions to the Report" but in the **Custom Permissions** dialog, enable only the **Read** permission, as shown in Figure 8–27.

Figure 8–27 Setting Custom Permissions on the Data Model



3. Repeat this procedure for any other referenced objects (subtemplates or style templates).

Delete Permissions

If the custom report inherited permissions that you want to remove from the report, you can delete permissions as follows:

1. Navigate to the object in the catalog and click **More** and then click **Permissions**. The **Permissions** dialog launches and the inherited permissions are shown.
2. Select the permission to delete and click the **Delete** toolbar button.

Task: Grant Permissions to the Oracle Enterprise Scheduler Job Definition

If this report is scheduled through the Oracle Enterprise Scheduler, execution permissions must also be granted to enable users belonging to submit the job. See Task: Grant Relevant Permissions in Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs."

8.2.8 Making Reports Available to Users in the Reports and Analytics Pane

To make a report available to users through the Reports and Analytics pane, map the report to the work areas of the user roles that need access. For the procedure to map reports to work areas, see "Define Application Toolkit Configuration" in the *Oracle Fusion Applications Common Implementation Guide*.

8.2.9 Enabling Reports for Scheduling from the Reports and Analytics Pane

To enable scheduling through the Reports and Analytics pane, configure the report properties:

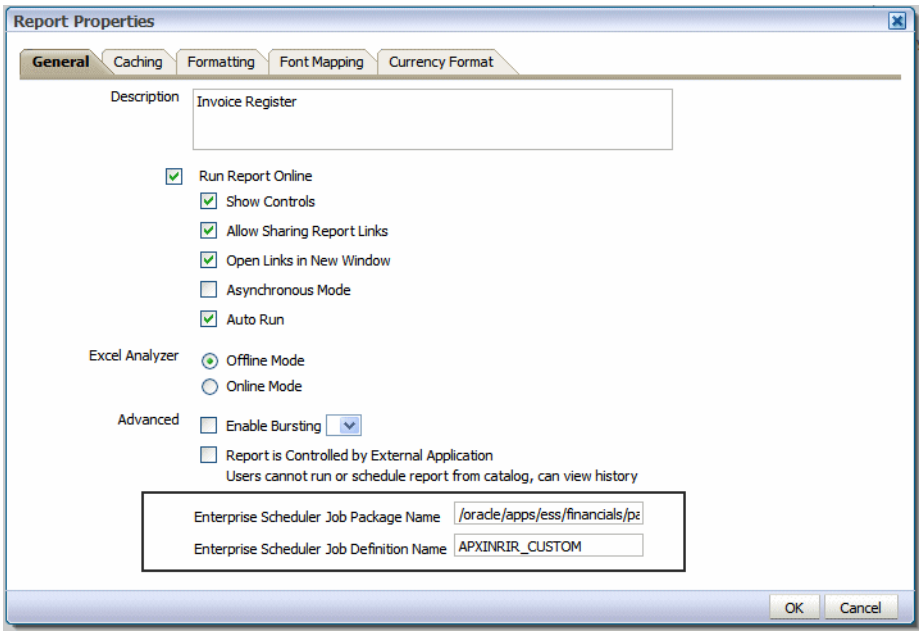
1. Navigate to the report in the Business Intelligence catalog and click **Edit.**
2. In the report editor, click **Properties**.
3. On the **Properties** dialog enter the following fields:
   - Enterprise Scheduler Job Package Name

     Enter the Path for the Job Definition. For example: /oracle/apps/ess/financials/payables/invoices/transactions/Jobs
   - Enterprise Scheduler Job Definition Name

     Enter the Name for the Job Definition. For example: APXINRIR

     The **Report Properties** dialog is shown in Figure 8–28:

Figure 8–28 Report Properties to Enable Scheduling Through the Reporting Pane



8.3 Customizing Analytics

This section describes how to use Oracle Business Intelligence Enterprise Edition to customize and extend analytics for Oracle Fusion Applications.

8.3.1 About Customizing Analytics

Analytics are analyses and dashboards built with Oracle Business Intelligence Presentation Services, based on objects in the Oracle BI repository. Analyses are queries based on real-time, transactional or operational data that provide answers to business questions. Dashboards provide personalized views of corporate and external information. A dashboard consists of one or more pages that contain content, such as analyses, links to Web sites, BI Publisher reports, and so on.

You can customize analyses using the Oracle BI Composer interface from within Oracle Fusion Applications. You can customize dashboards using Oracle Business Intelligence Enterprise Edition.

You can also customize objects in the Oracle BI repository (RPD) using the Oracle BI Administration Tool in either online or offline mode. Use online mode only for small changes that do not require running consistency checks. Running consistency checks against the full online repository can take a long time. Instead, make more complex changes that require consistency checks in offline mode against a project extract of the repository.

Table 8–4 provides guidelines for when to perform online and offline edits. See the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for full information about how to use the Administration Tool to edit RPD files.

Table 8–4 Guidelines for Online and Offline RPD Edits

Mode	Use This Mode For:	Example Use Cases	Example Operations Information
Online	■ Changes that do not require running a consistency check ■ Small changes that are required to fix things in a running system ■ Changes that need to be deployed quickly	■ Renaming Presentation Layer metadata ■ Reorganizing Presentation Layer metadata	1. Connect to the RPD in online mode. 2. Check out, modify, then check in the appropriate objects. 3. In a clustered system, restart all Oracle BI Servers except for the master server to propagate the changes. You can use the Cluster Manager in the Administration Tool to identify the master Oracle BI Server. 4. Reload metadata in Oracle BI Presentation Services by clicking the **Reload Files and Metadata** link from the Administration page.
Offline	■ Full-scale development or customization activities that require running consistency checks multiple times and iterating	■ Configuring Descriptive Flexfields and Key Flexfields for Oracle Business Intelligence ■ Customizing existing fact or dimension tables ■ Adding new fact or dimension tables	1. Copy the RPD from the production computer to the Windows development computer. 2. Open the RPD in offline mode and make the appropriate changes. 3. Upload the repository using Fusion Applications Control and restart all Oracle Business Intelligence system components.

8.3.1.1 What You Can Customize in Analytics

You can customize analyses and dashboards, as well as objects in the Oracle BI repository (RPD).

Customizations to analyses and dashboards result in changes to the Oracle BI Presentation Catalog. Be aware that some patches include updates to the Oracle BI Presentation Catalog. All new objects are preserved during the patch process; in addition, changes to existing objects are preserved when the patch does not include a new version of that object.

If you change an existing presentation catalog object and subsequent patches do include a new version of the object, the patch process detects and logs conflicts, and patching will stop. The catalog administrator must resolve any conflicts manually using Catalog Manager and then rerun the patch.

In the Oracle BI repository, you can create new repository objects such as physical columns, logical table sources, logical columns, and presentation columns. Be aware that some patches include updates to the Oracle BI repository. New objects are preserved during the patch process; in addition, changes to existing objects are preserved when the patch does not include a new version of that object.

If you change an existing object and subsequent patches do include a new version of the object, the Merge Wizard in the Administration Tool provides a method to merge the changes. For most typical customizations, the merge process is straightforward. The exception is when presentation columns have been moved across presentation tables; in this situation, it is important to plan ahead and track the changes carefully to ensure your changes are preserved during the merge.

See the "Oracle BI Applications Patching" chapter in the *Oracle Fusion Middleware Reference Guide for Oracle Business Intelligence Applications* for more information about patching the Oracle BI Presentation Catalog and Oracle BI repository.

8.3.1.2 Before You Begin Customizing Analytics

Before you customize analytics, ensure you have proper permissions for editing and creating Oracle Business Intelligence Presentation Catalog objects and understand how to set permissions in the catalog. For more information about setting permissions in the catalog, see the "Managing Objects in the Oracle BI Presentation Catalog" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*.

In addition, you must have the BIAuthor role to customize analytics (either explicitly granted, or inherited from another role).

Follow these guidelines when customizing analytics:

- When customizing referenced objects (such as embedded dashboards or targets of navigation actions), consider customizing them in place using "Save." Note that objects for Oracle Transactional Business Intelligence and Oracle Business Intelligence Applications provide conflict detection so that your customizations will not be overwritten during future patches.
- When customizing objects that are not referenced, consider using "Save As." You have the following choices when using Save As:
  - Existing folder structure (recommended)

    Saving to the existing folder structure extends the organization of your existing reports to include the custom reports. To use this approach, make sure that an Oracle BI administrator (a user with the BIAdmin role) grants Write permissions to the BIAuthor role for the given folders. Note that the reports inherit folder permissions that control which roles have Read and Write access.
  - New folders under Shared Folders

    To use this approach, an Oracle BI administrator must create these folders and then grant Write permission to the BIAuthor role, as well as Read permission to other application roles as needed.
  - My Folders

    Because nobody else can access My Folders, you do not typically save analytics to that location except for testing purposes.

8.3.2 Customizing Analytics

You can customize analytics from the Reports and Analytics pane in Oracle Fusion Applications.

Task: Customizing Analytics

To customize analytics, go to the Reports and Analytics pane and locate to the object you want to customize. For analyses, click the object and then select **Edit** to use the Oracle BI Composer to edit the object. For dashboards, click the object and then select **More** to go to the Catalog page in Oracle Business Intelligence Enterprise Edition.

For information on customizing analyses using Oracle BI Composer, see the "Using BI Composer to Work with Analyses" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*.

For information on customizing dashboards in the Catalog page, see the "Building and Using Dashboards" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*.

8.3.3 Customizing the Oracle BI Repository (RPD)

You can customize and extend the Oracle BI repository (RPD file).

Task: Create BI View Objects for Custom Fact and Dimension Tables

Whenever you create a custom fact or dimension table, you must create a BI view object for that table and incorporate it into the Oracle Fusion application before you can import it into the Oracle BI repository. To do this, follow these steps:

1. From a JDeveloper application workspace in the developer role, define the custom view object for the custom table. You must follow the view object guidelines for Oracle Transactional Business Intelligence as described in "Designing and Securing View Objects for Oracle Business Intelligence Applications" in the *Oracle Fusion Applications Developer's Guide*. For information about creating view objects from an application workspace, see Section 11.5, "Creating Custom Business Components."

   Tip: When you create the custom table, you must grant the necessary privileges (such as `SELECT`) to the FUSION_BI schema user in addition to the FUSION_RUNTIME schema user. Otherwise, queries against the new table will fail.

   See also Section 11.8, "Customizing and Extending the Oracle Fusion Applications Schemas" for guidelines on creating custom tables.

2. Create an application module (AM), as described in Section 11.5, "Creating Custom Business Components," and add the custom BI view object instance to the application module.
3. Create an ADF Library JAR for the custom artifacts as described in Section 11.13, "Deploying ADF Customizations and Extensions."
4. From a customization workspace, import the ADF Library JAR for the custom artifacts into the Oracle Fusion application and restart the host server for the application so that the customizations are picked up. For more information, see Section 10.2, "Customizing Oracle ADF Artifacts with JDeveloper."
5. Continuing in the customization role, nest the BI application module in the root application module as described in the "Defining Nested Application Modules" section in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
6. In the customization role, create a MAR file and load the MAR file using WLST commands or the Oracle WebLogic Server Administration Console as described in

Section 11.13, "Deploying ADF Customizations and Extensions," and restart the host server for the application so that the customizations are picked up.

Task: Modifying Existing Fact or Dimension Tables

In some cases, you might want to modify existing fact or dimension tables in the Oracle BI repository. For example, say you want to deploy Oracle Fusion Project Portfolio Management, but use the PeopleSoft Procurement application as a source. In this situation, you would set up a custom table in Oracle Fusion Applications that populates Commitments data from PeopleSoft. Then, you would need to change the Commitments fact table in the Oracle BI repository (RPD file) to point to the new custom table.

To accomplish the task described in this example:

1. Create a custom BI view object for the custom table and incorporate it into the application as described in Task: Create BI View Objects for Custom Fact and Dimension Tables.
2. Use the Import Metadata Wizard in the Administration Tool to import the new view object into the Physical layer of the RPD under the appropriate database object. Then, join the new view object to the existing dimension view objects. You must connect as the FUSION_APPS_BI_APPID user in the Select Data Source screen of the Import Metadata Wizard.

   See the "Importing Metadata from ADF Business Component Data Sources" and "Working with Physical Foreign Keys and Joins" sections in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information.
3. Create a new logical table source under the existing Commitment logical fact table, and map all metrics to the physical columns from the new view object. Then, deactivate the existing Commitments logical table source.

   See the "Managing Logical Table Sources (Mappings)" chapter in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information.

Using this approach, all Presentation layer metadata, analyses, and dashboards will continue to work with data coming from the new physical columns.

Task: Adding New Fact or Dimension Tables

In some cases, you might want to add new fact or dimension tables to your Oracle BI repository. Possible sources include custom tables in Oracle Fusion Applications, additional tables in the data warehouse, or new physical data sources.

To add new fact or dimension tables to your RPD:

1. For Oracle Fusion Applications sources, create a custom BI view object for the custom table and incorporate it into the application as described in Task: Create BI View Objects for Custom Fact and Dimension Tables.
2. Use the Import Metadata Wizard in the Oracle BI Administration Tool to import the new view object (for Oracle Fusion Applications) or physical table (for warehouse or other physical sources) into the Physical layer of the RPD. For non-warehouse physical sources, you must create a new connection pool as part of the import process. You must connect as the FUSION_APPS_BI_APPID user in the Select Data Source screen of the Import Metadata Wizard.

For more information, see the following resources in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)*:

- **Oracle Fusion Applications sources:** The "Importing Metadata from ADF Business Component Data Sources" section
- **Other physical sources:** The "Importing Metadata and Working with Data Sources" chapter

3. Define new logical dimensions and measures to extend the semantic model, and add physical and logical joins.

   See the "Working with Logical Tables, Joins, and Columns" and "Working with Physical Foreign Keys and Joins" sections in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information.

4. Add corresponding Presentation layer metadata.

   See the "Creating and Maintaining the Presentation Layer" chapter in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information.

Task: Changing How Metadata Is Displayed in Answers Reports

In some cases, you might want to change how the names of facts and dimensions in the Presentation layer appear in Answers reports, to comply with naming standards or for other reasons. Table 8–5 summarizes considerations for different use cases.

Table 8–5 Use Cases for Changing How Metadata Is Displayed

Use Case	For More Information
For warehouse sources, display names are typically externalized into a database table. To customize the names, you can change them in the externalized tables with no impact to the metadata itself. Note that for situations where display names are externalized into a database table, changing the names of Presentation layer objects in the RPD has no impact on the names displayed in Answers reports.	See the "Localizing Metadata Names in the Repository" section in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information about externalizing display names.
For Oracle Transactional Business Intelligence sources, display names are typically customized using UI hints (labels and tooltips) within Oracle Fusion Applications. Changing the UI hint name does not impact metadata. Note that for situations where display names are customized using UI hints, changing the names of Presentation layer objects in the RPD has no impact on the names displayed in Answers reports.	See the "Propagating Labels and Tooltips from ADF Business Component Data Sources" section in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information about how UI hints are propagated in the RPD.
For situations where Presentation layer names are not externalized or tied to UI hints, display names must be modified directly in the RPD. Existing reports will continue to work because the old names are stored as aliases.	See the "Renaming Presentation Columns to User-Friendly Names" section in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information.

Task: Reorganizing Presentation Layer Metadata

Note the following about reorganizing Presentation layer metadata in the RPD:

- Reordering presentation columns within a presentation table will not cause existing reports to break. When subsequent patches are applied, the new custom order is preserved when the patch does not include changes to the column order for that table.
- Moving presentation columns across different presentation tables can cause existing reports to break and is not recommended. If you do move presentation columns across tables, it is important to plan ahead and track the changes carefully.

Task: Configuring Descriptive Flexfields and Key Flexfields for Oracle Business Intelligence

You can use the Import Metadata Wizard in the Administration Tool to incrementally import flexfield changes to the Physical layer of the Oracle BI repository (RPD).

> **Tip:** See Chapter 5, "Using Flexfields for Custom Attributes" for information about changing flexfields. In particular, see "Task: Incorporate Custom Attributes into Oracle Business Intelligence" for information about enabling flexfields for BI.

In addition, you can use the Map to Logical Model screen of the Import Metadata Wizard to automatically propagate the flexfield changes to the Business Model and Mapping layer.

Finally, for Oracle BI Applications customers, you can configure and enable the BI Extender functionality to propagate flexfield changes to the data warehouse.

See the following sections in the *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)* for more information about these topics:

- "Using Incremental Import to Propagate Flex Object Changes"
- "Automatically Mapping Flex Object Changes to the Logical Model"
- "Using the BI Extender to Propagate Flex Object Changes"

Task: Moving RPD Changes to Production Systems

Typically, data source connection pool settings are different in production repositories. You can use the Oracle BI Server XML API to programmatically update these settings in the repository when moving changes to production systems. See the "Moving from Test to Production Environments" section in the *Oracle Fusion Middleware XML Schema Reference for Oracle Business Intelligence Enterprise Edition* for more information.

9

Customizing Security for Custom Business Objects

This chapter describes how to use CRM Application Composer to define and edit role-based security policies on custom business objects at runtime in certain Oracle Fusion Customer Relationship Management applications, limited to the Sales, Marketing, Customer Center, Trading Community Architecture, or Order Capture applications.

This chapter includes the following sections:

9.1 About Defining Security Policies

Security for Oracle Fusion Applications is configured to protect the data and business functions of the enterprise from unintended access. By default, new business objects and the web pages that display them are inaccessible to any user other than the user of CRM Application Composer. This means that when you create a security policy in CRM Application Composer, you specifically *grant* access to resources that would otherwise be protected. Additionally, security policies defined in CRM Application Composer are enforced on *all* the data records of the business object. Controlling access to individual data records is not supported for custom objects.

To enable access to the data records, you can use CRM Application Composer to create security policies for new business objects that they add to the following CRM applications:

- Sales (Partner Center)
- Marketing
- Customer Center
- Trading Community Architecture
- Order Capture

> **Note:** For information about how to define security policies for custom business objects in other Oracle Fusion applications, see Chapter 15, "Customizing Security for ADF Application Artifacts."

In Oracle Fusion Applications, the business object defines the available operations that may be performed over a particular set of data. The business object also encapsulates the data as business object instances, corresponding to data records from a database table. Typical operations are business functions like viewing, editing, or creating an instance of the business object. A security policy essentially needs to specify "who can perform what operations on what specific data."

Security policies in Oracle Fusion Applications provide *role-based access control* to the data records encapsulated by the business object, as well as the application artifacts, like web pages that interact with those data records. Role-based security ensures that the person creating the security policy does not require information about the individuals comprised by the enterprise at any given time. Rather, a security policy is always associated with a predefined role that end users are anticipated to fill when interacting with Oracle Fusion Applications.

For example, in a sales organization, duties such as `Manage_Accounts` and `Manage_Invoices` exist for roles, such as `Sales_Manager` or `Sales_Associate`. A security policy might give end users who belong to the `Sales_Associate` role the ability to view and edit the data records exposed by a particular business object, such as a customer invoice, but not to delete the records. Whereas, another security policy could grant end users who belong to the `Sales_Manager` role, the right to view, edit, and delete the same data records.

A security administrator for the enterprise completes the security configuration task by provisioning end users of the enterprise with one or more roles, based on the variety of duties the end user is expected to perform. The security policies defined for that role, in turn, confer to its member end users specific access right, or privileges.

> **Note:** Security-related configuration tasks such as configuring the enterprise identity store, configuring roles, and provisioning end user identities are not supported in CRM Application Composer. For details about security configuration, see the "Securing Oracle Fusions Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

In summary, a security policy for Oracle Fusion Applications considers the duties end users perform and then grants a role specific rights to:

- Access the page that supports the duty
- Access the specific data records, or instances of the business object, required to complete the duty
- Perform only those operations on that data required by the duty.

9.1.1 About the Implementation of Security Policies in CRM Application Composer

Although CRM Application Composer does not implement the security policy directly on the business object, the user interface focuses on business objects as a convenient way to represent and manage the actual security policies of the enterprise. Specifically, CRM Application Composer security policy screens simplify the task of choosing a

business object and then creating a complete security policy to grant the level of access needed by any role in Oracle Fusion Applications.

A complete security policy is one that controls access to the selected business object's operations, its instances, as well as the web pages that display the actual data records exposed by the business object. In the background, when you create a security policy for a custom business object and a role, CRM Application Composer interacts with the Oracle Fusion Applications security repositories to create or update the specific security artifacts that define the policy.

Specifically, the artifacts created in the Oracle Fusion security repositories by CRM Application Composer correspond to Oracle Fusion Data Security policies and Oracle Platform Security Service (OPSS) function security policies. A security policy that defines the level of access to the data records of the business object is known as a *data security policy*. A security policy that defines the level of access to the application resources that display the object is known as a *function security policy*. To completely specify access, both types of security policies must exist for the role.

While both types of security policies are conceptually similar, the repository for storing the security artifacts, as well as the representation of the artifacts, differ. CRM Application Composer defines the data security policy in Oracle Fusion Data Security database tables and defines the OPSS function security policy as hierarchically organized objects in an LDAP directory service.

For the user of CRM Application Composer, the distinctions of the type of security policy and the specific repository used to store them are not relevant. CRM Application Composer security policy screens do not label policies as data policies and function policies. In CRM Application Composer, the security policy screens hide these distinctions and instead allow you to focus on the business object. The security policy screen lets you view the business objects by name and modify the privileges granted to the various Oracle Fusion Applications roles to access the data records exposed by the business object and to access the web page created to display those records.

To understand the implementation details of data security and function security, you can read about the tools developers use to interact directly with the security repositories, as described in Chapter 15, "Customizing Security for ADF Application Artifacts."

9.1.2 What You Can Do With Security Policies at Runtime

After you create a custom business object and then create the web page to display the data records of the business object in CRM Application Composer, these application artifacts will be secured in Oracle Fusion Applications by default. This means end users will be denied access until you grant them access through a role-based security policy that you define.

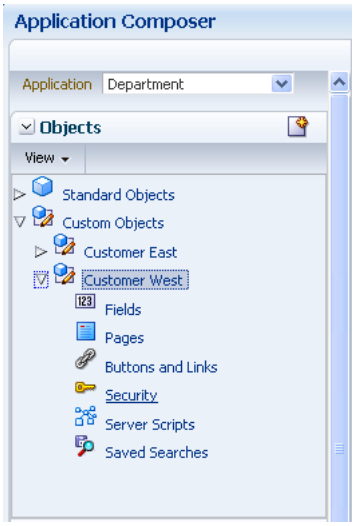
To enable access, you can use CRM Application Composer to create role-based security policies for new business objects that are added to the following CRM applications:

- Sales (Partner Center)
- Marketing
- Customer Center
- Trading Community Architecture
- Order Capture

Note: For information about how to define security policies for custom business objects in other Oracle Fusion applications, see Chapter 15, "Customizing Security for ADF Application Artifacts."

Figure 9–1 shows the Application Navigator in CRM Application Composer with the **Custom Object** list expanded to display the **Security** option for a custom business object. The **Security** option is displayed in the Application Navigator only for custom business objects. After you select this option, the object-centric security policy screen opens in CRM Application Composer and displays the policies for the expanded business object (as shown in Figure 9–1).

Figure 9–1 Navigator Displays Security Customization Option for Custom Objects in CRM Application Composer



Note: The term *custom object* is used in CRM Application Composer to distinguish business objects that you create from standard objects that are part of the Oracle Fusion Applications security reference implementation. Security policies for standard objects are not exposed in CRM Application Composer. For more information about configuring security for the reference implementation, see Section 15.2, "About Extending the Oracle Fusion Security Reference Implementation."

Because security policies provide role-based access, in CRM Application Composer a security policy specifies a role name, a custom business object name, and one or more privileges that specify the level of access granted to the role. As described in Section 9.1.1, "About the Implementation of Security Policies in CRM Application Composer," when you use the security policy screens in CRM Application Composer, you will not need to create the underlying security artifacts; the tool will create those for you in the appropriate Oracle Fusion security repository.

The security screens in CRM Application Composer let you grant and revoke access privileges to roles in one of two formats.

Figure 9–2 shows the screen that lets you display a single *custom object* and view all the roles and the level of access that each one defines for that custom business object. In

this example, the object-centric security screen displays the custom business object **CustomerWest** and shows a default security policy defined on the **CRM Application Administrator Duty** role (note the default role to use for testing security can be configured for the Sales, Marketing, Customer Center, Trading Community Architecture, and Order Capture applications). CRM Application Composer creates this security policy by default so that you can view and customize the custom object in CRM Application Composer. In this screen, no other policies have been defined.

Figure 9–2 Object-Centric Security Policy Screen: Viewing and Modifying All Security Policies for a Given Custom Business Object

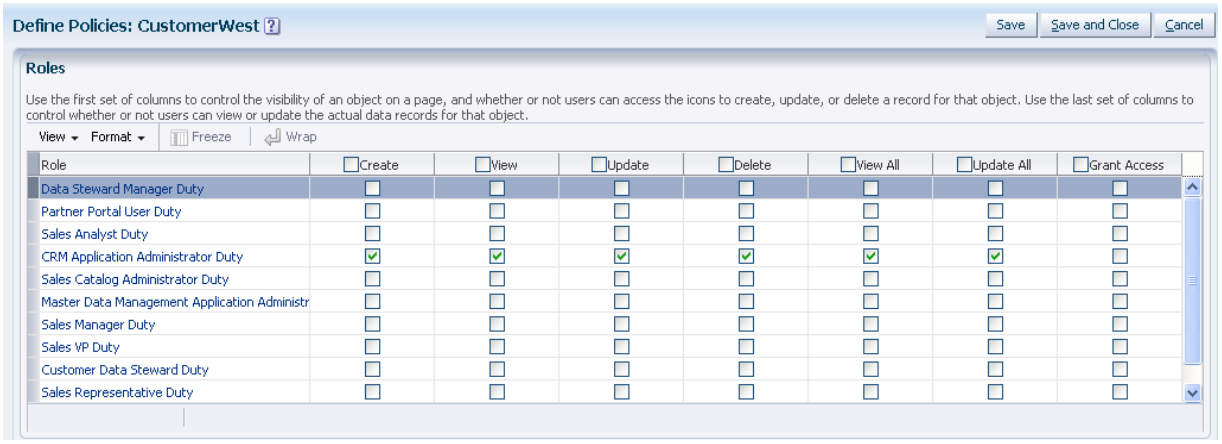
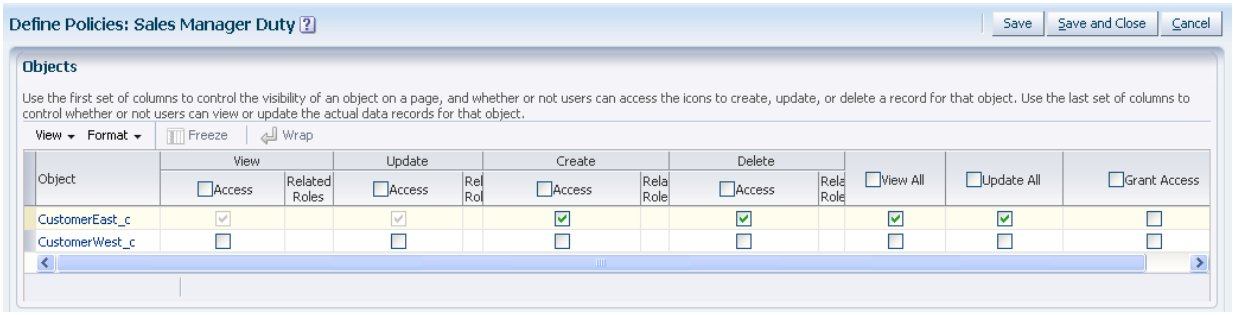


Figure 9–3 shows the alternative screen you can open to display a single *role* and view the level of access it grants to *all* the custom objects in the Sales, Marketing, Customer Center, Trading Community Architecture, or Order Capture applications. In this example, the role-centric security screen displays the **Sales Manager Duty** role with access granted to the **CustomerEast** business object.

Figure 9–3 Role-Centric Security Policy Screen: Business Viewing and Modifying All Security Policies for a Given Role



The selection choices presented by the columns of both security policies screens (the object-centric screen and the role-centric screen) have the same meaning:

- The first four columns **View**, **Update**, **Create**, and **Delete** correspond to the privileges that confer rights to the web page that you create to display the data records of the custom business object.

  The column selections define a function security policy in the LDAP security repository and determine whether the end user may view the web page, and then, assuming the page is displayed, whether the buttons that enable operations to edit

a data record, create a new data record, or delete a data record will themselves appear enabled or disabled (grayed out) in the page.

At runtime, in the Oracle Fusion application, the end user may have the right to view the data records displayed by the web page, but unless their role also confers the right for example to edit the page, the user interface displays the page with the Edit button disabled to prevent this operation. Until **View**, **Update**, **Create**, and **Delete** columns are selected for a role, no function security policy exists in the LDAP security repository and the custom business object's web page remain protected by default, thus preventing all end users provisioned with that role from accessing the page and, consequently, also the data.

- The next two columns, **ViewAll** and **UpdateAll**, confer rights to view and manipulate the data records of the custom business object in the web page.

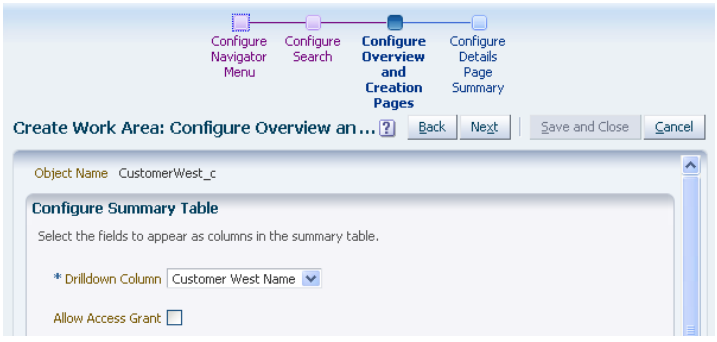
  The column selections define a data security policy in the Oracle Fusion Data Security security repository and determine whether the end user may view the data records and, assuming the records are displayed in the web page, whether the end user has the rights to edit or delete the data records exposed by the custom business object. When you select **UpdateAll**, as a side-effect, CRM Application Composer automatically enables the corresponding function security privileges to give the end user the right to select the Edit buttons in the web page. Until **ViewAll** or **UpdateAll** columns are selected for a role, no data security policy exists in the repository and the data records remain protected by default, thus preventing all end users provisioned with that role from accessing the data.

- The last column, **Grant Access**, enables a runtime security configuration feature that gives end users the ability to share their security entitlements with another end user.

  When you enable "sharing" of entitlements for a specific custom business object, you allow one user to confer their privileges to another end user. The **Grant Access** option enables the feature at the level of the business object so it will be effective in any page that displays the data records of the custom business object.

Figure 9–4 shows an alternative way to enable the entitlement sharing feature. When you use the page creation screen that you display for the custom business object, the option labeled **Allow Access Grant** lets you to enable the feature for the custom business object at the level of a single page.

Figure 9–4 Create Work Area Flow: Enabling Entitlement Sharing



When you use CRM Application Composer, you can optionally elect to do your work in a sandbox, and after you publish your sandbox, all business objects, pages, and the security policies you define become part of the running application. When you are ready to edit security policies, you can initiate the security sandbox setup operation. This setup operation will duplicate the schema for Oracle Fusion Data Security tables

and is necessarily a lengthy one that must be allowed to complete before customization can begin. After you complete the customizations, published security policies will be merged into the Oracle Fusion security policy repository as part of the native application and they will overwrite any previous customizations.

> **Note:** Because inconsistencies can result when multiple users edit the security policies associated with the same object in different sandboxes, users may coordinate so they avoid customizing the same object concurrently. For more information about runtime customization and the sandbox, see Section 2.2, "Using the Sandbox Manager."

In summary, using CRM Application Composer, you can perform these tasks to define security policies for a custom business object:

- Grant and revoke access privileges made to specific roles (such as Sales Manager or Sale Representative). For details about this task, see Section 9.2, "Defining Security Policies for Custom Business Objects."
- Enable end users to elevate the privileges of other end users by conferring their own rights to view, edit, or delete individual data records. For details about this task, see Section 9.3, "Enabling End User Instance-Level Security Customization."

9.1.3 What You Cannot Do With Security Policies at Runtime

CRM Application Composer does not expose the underlying implementation details of the actual security policies created in the Oracle Fusion security repositories. Other tools in the Oracle Fusion Applications environment provide the enterprise security administrator and other appropriately authorized end users with complete control over the creation and viewing of those security artifacts.

Because CRM Application Composer provides an abstracted view of the security policy implementation, it also limits your ability to edit security policies. For example, a developer may need to control access to specific records of the custom business object that they implement, and they may use other tools to interact directly with the security repository to make this type of customization. Whereas, in CRM Application Composer, when you grant access privileges to a given role for a custom business object, CRM Application Composer creates a global instance set that specifies all data records of the business object. Therefore, no capability exists currently in CRM Application Composer to stripe individual data records with specific access privileges.

While you can create role-based security policies using CRM Application Composer, the following are more advanced use cases for which you will need to use development and administration tools:

- Create custom roles or modify the role inheritance hierarchy provisioned by Oracle Fusion Applications. In those cases where you will need to use a custom role, consult a system administrator. Only a system administrator may create a new role, and optionally include the role in an existing role hierarchy. For information about creating application roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.
- Define or edit a security policy contained in an application that is not Sales, Marketing, Customer Center, Trading Community Architecture, or Order Capture. In those cases, you will need to use other tools instead of CRM

Application Composer to define your security policies. For more information, see Chapter 15, "Customizing Security for ADF Application Artifacts."

- Define or edit a data security policy for the standard business objects defined by any Oracle Fusion application. CRM Application Composer supports defining security for custom business objects only. Only an authorized an security administrator can manage the Oracle Fusion security reference implementation where they use other tools to edit policies for standard objects. For more information, see Section 15.2, "About Extending the Oracle Fusion Security Reference Implementation."
- Define data security policies on individual business object instances or groups of instances. Security policies that you define in CRM Application Composer are enforced on the all the data records of the business object (referred to as a *global instance set*). Controlling access to individual data records is not supported on custom objects. Only customization developers and an security administrators can manage data security policies. For more information, see Section 15.3.4, "Scenarios Related to Extending and Securing Data Model Components."
- Enforce column-level security. Security policies that you define in CRM Application Composer are enforced on the data records (or rows) of the business object. Controlling access to columns of data requires using Oracle JDeveloper to create a customization workspace for the application. For more information, see Section 15.3.4, "Scenarios Related to Extending and Securing Data Model Components."
- Define function security policies on individual application artifacts. Security policies that you define in CRM Application Composer will automatically be enforced on the application resources that reference the custom business object based on the View, Edit, Update, and Create privileges you select. Enforcing security on specific application resources, such as a customized task flow and its web pages or components in a web page, requires using JDeveloper to create a customization workspace for the application. For more information, see Section 15.3.5, "Scenarios Related to Extending and Securing User Interface Artifacts."

9.1.4 Before You Begin Customizing Security

Before you begin customizing security in CRM Application Composer, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can begin customizing security:

- Obtain the privileges needed to define security policies in CRM Application Composer.

  If you will be defining or editing security policies in Oracle Fusion Applications, you will need the correct privileges. When you have the correct privileges, CRM Application Composer will give you access to the security customization user interface. Please contact your security administrator for details.

- Optionally set up a sandbox.

  CRM Application Composer can use sandboxes to manage your customizations. For more information, see Section 2.2, "Using the Sandbox Manager."

Note: Because setting up the security sandbox requires duplicating the schema for Oracle Fusion Data Security tables, this will always be a lengthy operation in CRM Application Composer. Be sure to allow sufficient time for the process to complete and do not to terminate it early. You may want to defer customizing security and enabling the security sandbox until you are sure that you need to make customizations.

- Create the business object, as needed.

  Unless the business object appears in the Application Navigator of CRM Application Composer, you will not be able to define security policies using CRM Application Composer. You can create custom business objects for Sales, Marketing, Customer Center, Trading Community Architecture, or Order Capture applications in CRM Application Composer. For more information about creating business objects for these applications, see Task: Create Custom Objects in Section 4.4, "Creating Custom Objects." Business objects for all other applications must be created by a developer, administrator, or security manager, as described in Chapter 15, "Customizing Security for ADF Application Artifacts."

- Consult the system administrator to create custom roles, as needed.

  The access privileges specified by security policies you define for the custom business object are granted to application roles. Oracle Fusion Applications defines a large number of application roles based on the duties of its member end users. When an application role does not exist that adequately describe the duties pertaining to the custom business object, then a custom application role will need to be created. In those cases where you will need to use a custom role, consult a system administrator. Only a system administrator may create a new role, and optionally include the role in an existing role hierarchy.

9.2 Defining Security Policies for Custom Business Objects

Until you define a security policy for a custom business object, the data records exposed by that business object will be protected and end users will not have access to the data. When you want to make the custom business object accessible, you define a security policy using CRM Application Composer.

The security policy that you define in CRM Application Composer consists of the following access privileges that you select to control access to the custom business object by end users provisioned to particular roles:

- Grant **View**, **Update**, **Create**, and **Delete** privileges on the web page that displays the data records. Minimally, you must grant the **View** privilege to allow the end user to open the web page. You can also grant **Update**, **Create**, and **Delete** privileges to enable the buttons in the web page that the end user clicks to initiate these operations on the data records.

  Note: If an **Update**, **Create**, and **Delete** privilege is not granted, the corresponding button will appear grayed out (disabled) in the web page.

- Grant **ViewAll** and **UpdateAll** privileges on the data records themselves. Minimally, you must grant the **ViewAll** privilege to populate the web page with the data records for the end user. You can also grant the **UpdateAll** privilege to enable the end user to modify the data contained in any data record exposed by the custom business object.

  Note: When you grant the **UpdateAll** privilege, CRM Application Composer automatically grants the privileges for the edit operation button. You can enable or disable individual buttons (including edit, new, and delete) in the web page and

thereby deny end users access to individual operations on the data record by selecting or deselecting the corresponding **Update**, **Create**, and **Delete** privilege.

Before you begin:

Create the business object. You can only define security policies on custom business objects that you create in CRM Application Composer. The custom business object must exist before you define the security policy. For more information, see Task: Create Custom Objects in Section 4.4, "Creating Custom Objects."

Task: Grant View and Update Access to Multiple Roles Using the Object-Centric User Interface

The data records exposed by a custom business object can be accessed by members of more than one application role. You can grant access privileges to the view, update, create, or delete operations for a particular custom business object for each application role. In the navigator for CRM Application Composer, you select the custom business object and then you click **Security**. In the Define Policies screen, you grant access privileges to any of the displayed application roles for the previously selected custom business object as follows:

- When you want to grant view-only access, you select only the **View** and **ViewAll** privileges.
- When you want to grant view and update access, you select the **ViewAll** and **UpdateAll** privileges.

  After you select the **UpdateAll** privilege, the security policy screen automatically displays the **Update** privilege as selected.
- When you want to allow or deny access to individual operations to edit, create, or delete data records, you can select or deselect the corresponding **Update**, **Create**, and **Delete** privilege and leave the **UpdateAll** privilege selected.

For more information, see the "Securing Custom Objects: Explained" topic in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Grant View and Update Access to a Specific Role Using the Role-Centric User Interface

Members of an application role may have access to multiple custom business objects. You can grant access privileges to view, update, create, or delete operations for each of the custom business objects for a particular application role. In CRM Application Composer, you select **Security Roles** from the Common Setup panel. In the Security Roles screen, you select a role and then you click the **Define Policies** button. In the Security Policies screen, you grant access privileges to any of the displayed custom business objects for the previously selected role as follows:

- When you want to grant view-only access, you select only the **View** and **ViewAll** privileges.
- When you want to grant view and update access, you select the **ViewAll** and **UpdateAll** privileges.

  After you select the **UpdateAll** privilege, the security policy screen automatically displays the **Edit** privilege as selected.
- When you want to allow or deny access to individual operations to edit, create, or delete data records, you can select or deselect the corresponding **Update**, **Create**, and **Delete** privilege and leave the **UpdateAll** privilege selected.

For more information, see the "Securing Custom Objects: Explained" topic in the *Oracle Fusion Applications CRM Extensibility Guide*.

9.3 Enabling End User Instance-Level Security Customization

In certain situations one end user may require temporary access to the data records of another end user. In this scenario, Oracle Fusion Applications supports a runtime security customization feature that lets end users elevate the privileges of another end user by conferring their own rights to view, edit, or delete individual data records. Because each data record corresponds to a business object instance, this type of runtime customization is known as *instance-level security*.

In CRM Application Composer, this end user security configuration feature is enabled on the custom business object during page customization. At runtime, the page that displays the data records of the enabled custom business object will display a **Manage Permissions** button that opens a dialog that displays the list of end users to whom additional privileges may be granted. The privileges the dialog displays will be limited to the privileges available to the conferring end user, as defined in CRM Application Composer for the user's provisioned roles and the currently the displayed custom business object.

Additionally, to complete the configuration of this runtime security customization feature in CRM Application Composer, you must grant sufficient privileges to the roles which may be conferred a higher level of access by another end user in order to enable the Edit and Delete buttons in the user interface so that the user with elevated privileges can invoke edit or delete operations on the data record.

Before you begin:

Create the appropriate security policy on the custom business object for the conferring end user. The end user can only confer the access privileges to a data record that they already possess. This requirement means that the conferring end user's role must have the **ViewAll** or **UpdateAll** privileges granted in CRM Application Composer to be able to confer the right to view, edit, and delete a particular data record.

Create the appropriate security policy on the custom business object to be accessed by the target end user. At runtime, in Oracle Fusion application, the end user to whom the rights are granted must also have the privileges needed to select the buttons the web page displays to invoke the create, edit, or delete operations on the data records of the custom business object. This requirement means at least one role of the target end user must have the **View**, **Update**, and **Delete** privileges granted for the specific custom business object in CRM Application Composer. Otherwise, without these privileges, the Oracle Fusion application will display the web page with the Edit and Delete buttons grayed out (disabled) for the business object.

Task: Enabling End Users to Elevate the Access Privileges of Other End Users for a Business Object in a Specific Page

The data records of a custom business object represent instances of the object. You can enable a runtime security configuration feature that allows one end user to elevate the rights of another end user to access individual business object instances displayed in a specific page. Instance-level security lets one end user confer their own access privileges to other end users of the Oracle Fusion application in the enterprise. In the navigator for CRM Application Composer, to enable this runtime security configuration feature, you select the custom business object and then you click **Page**. In either of the **Pages** panels, you click through the page creation workflow until you reach the **Configure Landing and Creation Pages** task, and you then select **Allow**

Access Grant. For more information, see the "Securing Custom Objects: Explained" topic in the *Oracle Fusion Applications CRM Extensibility Guide*.

Task: Enabling End Users to Elevate the Access Privileges of Other End Users for a Business Object on Any Page

The data records of a custom business object represent instances of the object. You can enable a runtime security configuration feature that allows one end user to elevate the rights of another end user to access individual business object instances when they appear in any page. Instance-level security lets one end user confer their own access privileges to other end users of the Oracle Fusion application in the enterprise. In the navigator for CRM Application Composer, to enable this runtime security configuration feature, you select the custom business object and then you click **Security**. In the object-centric Define Policies screen, you select **Grant Access** for the desired application role. Alternatively, you can use the role-centric Define Policies screen to enable the same runtime security configuration feature. For more information, see the "Securing Custom Objects: Explained" topic in the *Oracle Fusion Applications CRM Extensibility Guide*.

9.4 Preventing Corrupted Security Policies in CRM Application Composer

CRM Application Composer creates a variety of security artifacts, which together provide security for the CRM application. After you define or edit a security policy in CRM Application Composer, authorized developers or security managers may be able to access the security repository and view the security policy. However, in the current release of Oracle Fusion Applications, even browsing the security policies for custom objects in tools such as Oracle Authorization Policy Manager or Oracle Fusion Functional Setup Manager may corrupt the security artifacts created in CRM Application Composer.

Important: Security managers must not use Oracle Authorization Policy Manager or Oracle Fusion Functional Setup Manager to browse or edit the security policies that you create in CRM Application Composer. Security policies created for custom objects, must therefore only be edited within CRM Application Composer.

Part III

Developer Customizations and Extensions

Part III contains the following chapters:

10

Using JDeveloper for Customizations

This chapter describes how to configure JDeveloper for implementing customizations in Oracle Fusion applications. It also describes how to customize Service-Oriented Architecture (SOA) composites with JDeveloper, including setting up the JDeveloper workspace and composite project for Oracle Metadata Services (MDS) Repository customization, customizing the composite, and customizing SOA resource bundles.

This chapter includes the following sections:

10.1 About Using JDeveloper for Customization

You use JDeveloper when you need to customize or create business objects or security outside of CRM applications, or when you need to make more sophisticated changes, like changes to SOA composites, Oracle Enterprise Scheduler Service jobs, BPM project templates, or embedded help. While you use JDeveloper to both customize existing standard objects and to create new custom objects, the procedures you use for each are different.

New custom objects created in JDeveloper are not saved into the MDS Repository, and so are done in a standard application workspace using the Oracle Fusion Applications Developer role. However, when you customize standard objects, those customizations *are* saved into the MDS Repository, and so must be done using the Oracle Fusion Applications Administrator Customization role. Doing the customizations using the customization role ensures that your changes are saved to the upgrade-safe MDS Repository, and not written directly to the standard object. In the future, when you patch or upgrade your Oracle Fusion Applications, your customizations held in these metadata files will not be touched, and so, you will not have to redo them. For more information about customizations and the MDS Repository, see Chapter 1, "Customizing and Extending Oracle Fusion Applications."

When customizing ADF artifacts, you create a special customization application workspace, using the developer role. This workspace includes a connection to a deployed Oracle Fusion Applications environment (typically a test environment), which allows you to import the artifacts you want to customize into your workspace. This customization workspace is automatically configured to work within Oracle Fusion Applications, so that when you test and deploy your customizations, they will behave as though they were native Oracle Fusion Applications. When customizing SOA composite applications, you create a SOA Composite application workspace in the developer role.

After the workspace is created, you switch roles to the customization role and import the ADF artifact or the SOA archive you want to customize. You then make your customizations to the imported artifact. After completion, you package and deploy the artifacts in the workspace to the Oracle Fusion Applications environment.

Often, you will need to perform both customizations (customizing an existing standard object) and extensions (creating a new object). For example, say you want to create a new entity object and expose that new object in an existing application module. First, because you are creating a new custom entity object, you would create a standard application workspace and then create your entity object. After completion, you would package the workspace as an ADF Library, and place it into the exploded EAR directory for your test environment. Next, you would create a customization application workspace, and import both the new entity object library and the library that contains the application module to which you need to add the entity object. After both are imported, you log in using the customization role and make the customizations to the application module. After customizations are complete, you would deploy the customizations to the test environment.

10.1.1 About Customizing Oracle ADF Artifacts

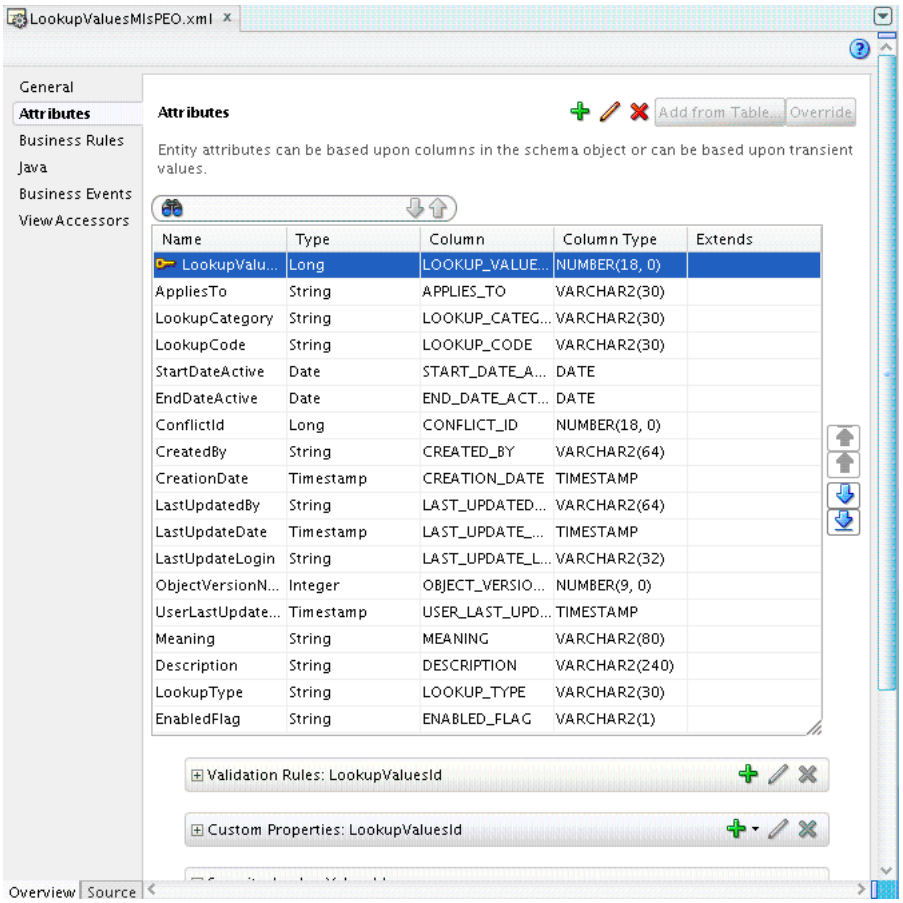
Oracle Fusion applications are built using Oracle ADF artifacts on Oracle Fusion Middleware, including the following:

- Application modules: An application module is the transactional component that UI clients use to work with application data. It defines an updateable data model along with top-level procedures and functions (called service methods) related to a logical unit of work related to an end-user task.
- Entity objects: An entity object represents a row in a database table and simplifies modifying its data by handling all data manipulation language (DML) operations for you. It can encapsulate business logic to ensure that your business rules are consistently enforced. You associate an entity object with others to reflect relationships in the underlying database schema to create a layer of business domain objects to reuse in multiple applications.
- View objects: A view object represents a SQL query and simplifies working with its results. You use the SQL language to join, filter, sort, and aggregate data into the shape required by the end-user task being represented in the user interface. This includes the ability to link a view object with other view objects to create master-detail hierarchies of any complexity. When end users modify data in the user interface, your view objects collaborate with entity objects to consistently validate and save the changes.
- Task flows: Task flows define the flow of control throughout an application. They also can be included in a page as a region, where users can navigate through a series of page fragments, without leaving the original page.
- JSPX pages and page fragments: The view layer of Oracle Fusion Applications consists of a small number of pages per application. These pages then contain task flows, which in turn contain a number of page fragments.

For more information about Oracle ADF components, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

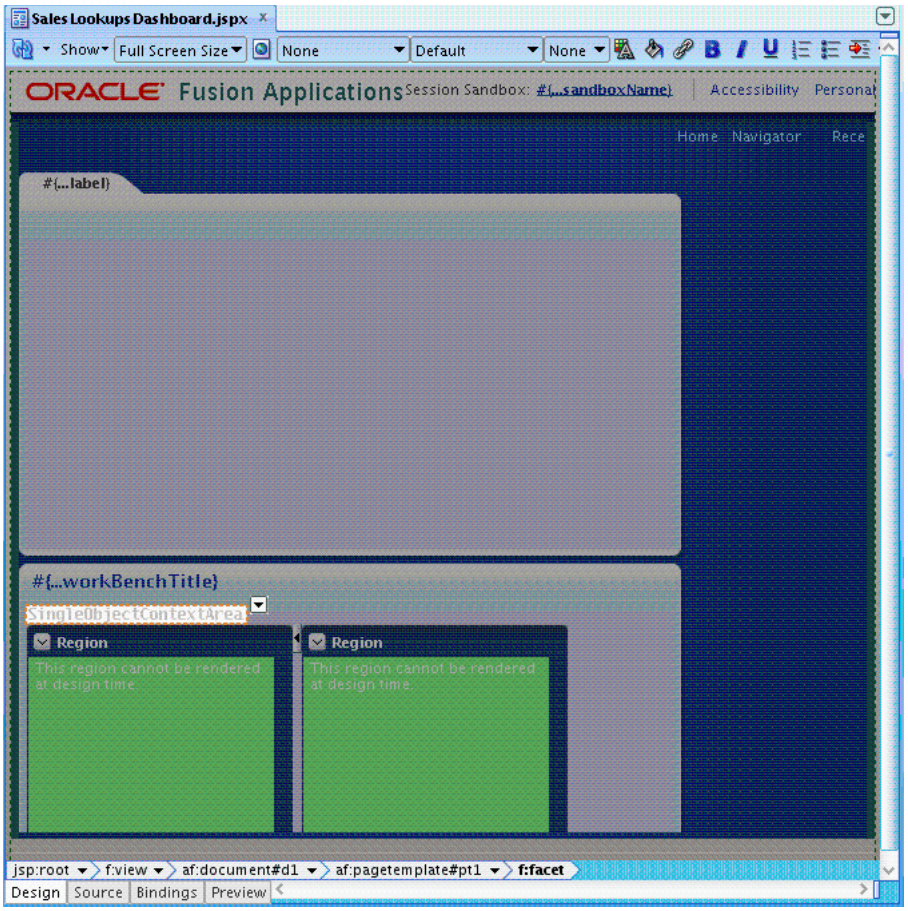
When you customize Oracle ADF artifacts, you usually work in an overview editor that allows you to make your customizations declaratively. For example, Figure 10–1 shows the editor for an entity object. Among other things, you can set validation or change how the UI displays the data.

Figure 10–1 Overview Editor for Entity Object



For JSP pages, you work in a WYSIWYG environment using the Design tab in the editor window, as shown in Figure 10–2.

Figure 10–2 Design Editor for JSP Pages



10.1.2 About Using JDeveloper to Customize SOA Composites

Oracle Fusion applications are built using SOA composite artifacts on Oracle Fusion Middleware, which include the following:

- Service components: Service components implement the business logic or processing rules of a SOA composite. Available service components include the following:
  - BPEL processes that enable you to integrate a series of business activities and services into an end-to-end business process flow.
  - Business rules that enable you to create business decisions in your business process flow based on rules.
  - Human tasks that enable you to create human workflows that describe the tasks for users or groups to act upon as part of an end-to-end business process flow. You use the graphical interface tool Oracle BPM Worklist to act upon the tasks during runtime.
  - Mediators enable you to define services that perform message and event routing, filtering, and transformations within the SOA composite.
- Binding components: Binding components establish the connection between a SOA composite and the external world. There are two types of binding components:

  - Services provide the outside world with an entry point to the SOA composite. The WSDL file of the service advertises its capabilities to external applications. These capabilities are used for contacting the SOA composite components. The binding connectivity of the service describes the protocols that can communicate with the service (for example, SOAP/HTTP or a JCA adapter).
  - References enable messages to be sent from the SOA composite application to external services in the outside world.
- Wires connect services, service components, and references into a complete SOA composite.

For more information about SOA composites, see the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

10.1.3 Before You Begin Using JDeveloper to Customize

Before you use JDeveloper to customize, be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." Also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can use JDeveloper to customize:

- The application you are customizing must be deployed to a test environment, and you must have access to the exploded EAR directory for that application.
- Install JDeveloper and set up your development environment. Before you can implement customizations using JDeveloper, you must create a workspace that imports the necessary parts of the application you want to customize. For more information, see the "Setting Up Your Development Environment" chapter in the *Oracle Fusion Applications Developer's Guide*.

Note: Before you can use JDeveloper to customize your application, JDeveloper must have access to the customization layers for the application. To enable JDeveloper to see the customization classes that define the customization layers, use the `-Dide.extension.extra.search.path` VM option, as described in the "Adding Customization Extension Bundles to the jdev.conf File" section of the *Oracle Fusion Applications Developer's Guide*.

For information about locating the JAR files containing the product-specific customization classes, see the product-specific documentation from Oracle Enterprise Repository for Oracle Fusion Applications. You can also use the steps in the "Adding Customization Extension Bundles to the jdev.conf File" section of the *Oracle Fusion Applications Developer's Guide* to locate the JAR files.

10.2 Customizing Oracle ADF Artifacts with JDeveloper

To customize ADF artifacts, you first create a customization application workspace, using the **Oracle Fusion Applications Developer** role in JDeveloper. After the workspace is created, you exit JDeveloper and then reenter, using the **Oracle Fusion Applications Administrator Customization** role and import and customize your artifacts.

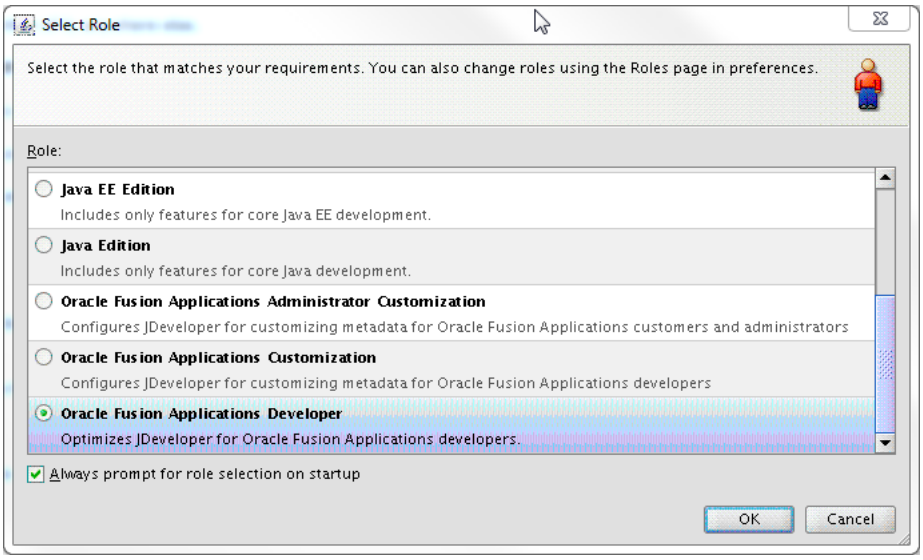
10.2.1 Creating the Customization Application Workspace

You need to set up a customization workspace in JDeveloper to create the application that will hold your customizations.

To create the customization application workspace

1. Start JDeveloper using the Oracle Fusion Applications Developer role, as shown in Figure 10–3.

Figure 10–3 Oracle Fusion Applications Developer Role



2. In JDeveloper, choose **File > New** to open the New Gallery. In the New Gallery, select **Applications > Fusion Applications Customization Application**.
3. In the Step 1 page of the FA Customization Application dialog, enter the following and click **Next**:
   - **Application Name** and **Directory**: This will be the name and location of your customization application, and can be anything you like.
   - **Fusion Database**: Enter the connection to your Fusion database.
   - **Application Package Prefix**: This can be anything, but must not start with `oracle`.
   - **Deployed Application Ear:** Browse to the exploded ear for the application you want to customize.
   - **Policy Store Security Information**: Browse to the exported `jazn-data.xml` file. First you must export all predefined function security policies of the application that you are customizing into a `jazn-data.xml` file. For details about how to export the application policy store, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

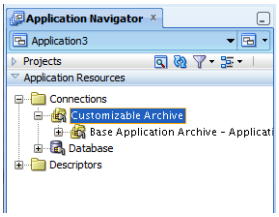
     For information about security customization, see Chapter 15, "Customizing Security for ADF Application Artifacts."
4. Complete the wizard by changing any default settings as needed.

After you complete the wizard, an application with a project is created for you. This application is configured to be the same as a deployed Oracle Fusion application. For

example, it is connected to the same database, same metadata repository, and has similar `web.xml`and `weblogic.xml` settings. This configuration allows it to work correctly when deployed into your Oracle Fusion Applications environment, and also ensures that when you test your customizations locally in the JDeveloper integrated Oracle WebLogic Server, the customizations will behave as they will in the full test environment.

JDeveloper also creates a connection to the exploded EAR directory named **Customizable Archive**, which is accessible from the Application Resources panel of the Application Navigator. Figure 10–4 shows a connection to the exploded EAR directory for an application.

Figure 10–4 Application Resources Connection to Exploded Ear Contents



10.2.2 Determining Which ADF Artifacts You Need to Customize

Most often, the customizations you want to make will be surfaced on an existing page. For example, say you want to add a field to a page. So, you first need to identify the page to customize, which may actually be a page fragment within a task flow. You then need to identify which business objects you'll need to customize to add the field.

The easiest way to identify which artifacts you need to customize is to follow this path:

1. In a runtime environment, access the page you want to customize and open it in the Source view of Page Composer. The page's structure is displayed, and from here, you can identify the page name, or if the customization is actually on a page fragment with in a task flow, you can identify the task flow name. For more information about using Page Composer, see Chapter 3, "Customizing Existing Pages."
2. If you need to customize a page fragment (`.jsff`) file within a task flow, from Page Composer, click **Manage Customizations** to open the page in the Manage Customizations dialog. From here, you can identify the `.jsff` file name.
3. In JDeveloper, after you have created a connection to the exploded EAR directory, you can use the Filter Customizable Archive dialog to search for the `.jsff` file or the task flow file.
4. Right-click the file and choose **Customize** to import the file and open it in JDeveloper.
5. Right-click the file, and choose **Go to Page Definition**.

   The page definition file will show you the view objects being used by the components on the page to return the data.
6. Open the view object in JDeveloper.

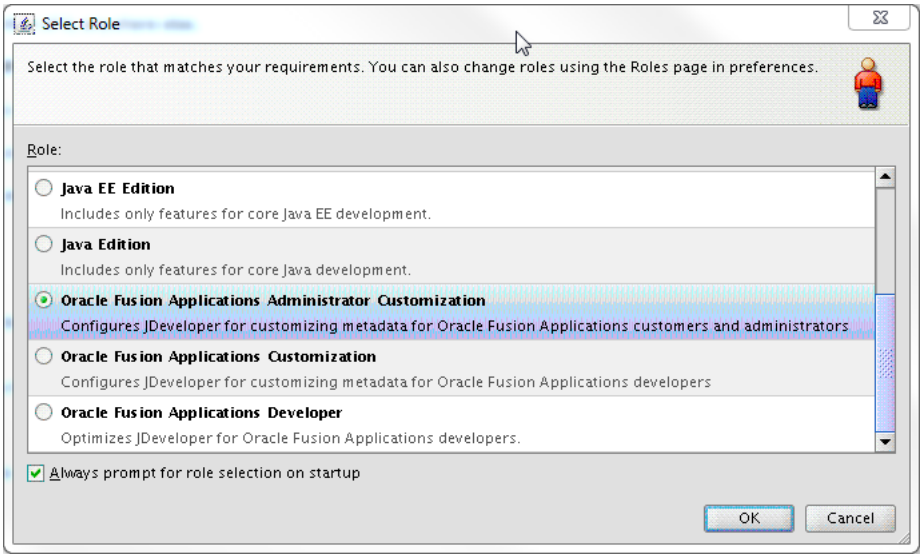
   The view object can be customized, or if needed, you can identify the associated entity object and customize that. You can also identify the application module from here.

10.2.3 Customizing the Artifacts

You need to switch to the Customization Developer role before you can begin customizing.

1. Restart JDeveloper and select the **Oracle Fusion Applications Administrator Customization** role, as shown in Figure 10–5.

Figure 10–5 Oracle Fusion Applications Administrator Customization Role



2. In the Application Resources panel, expand **Connections**, and then **Customizable Archive**.

3. To locate the artifact you want to customize, right-click **Base Application Archive** and choose **Filter**.

   For help in determining which artifacts you need to customize, see Section 10.2.2, "Determining Which ADF Artifacts You Need to Customize."

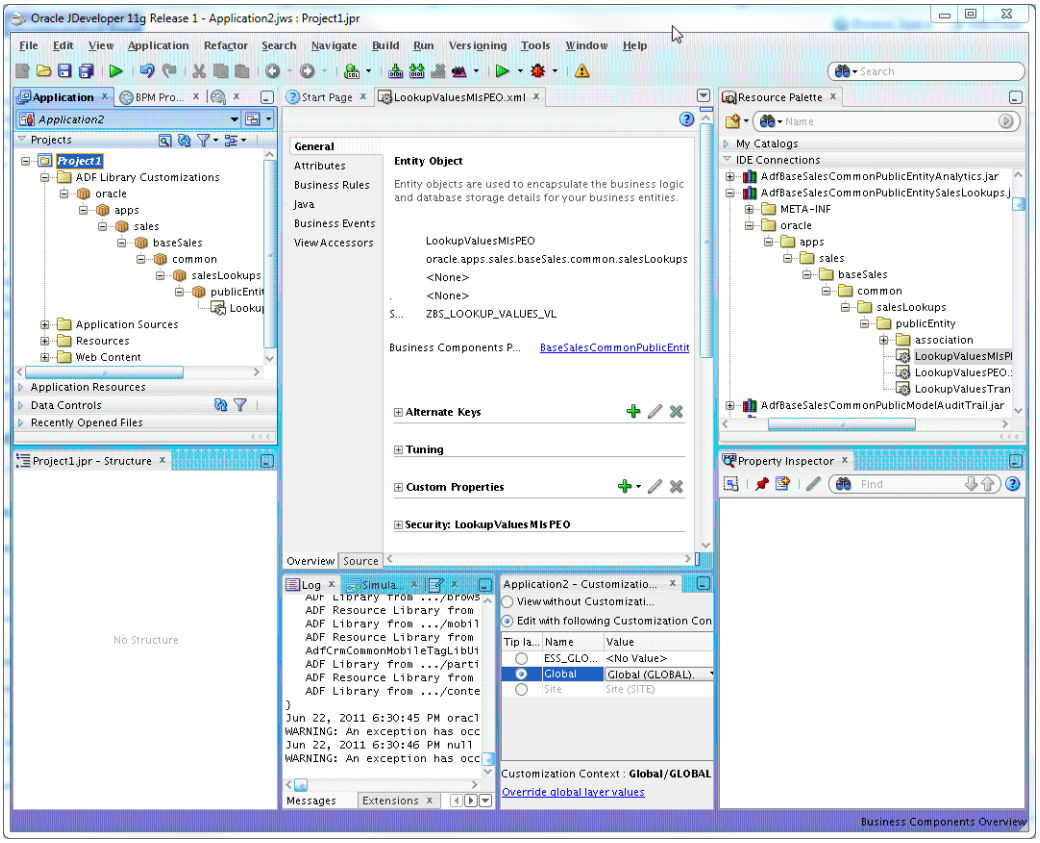
4. In the Filter Customizable Archive dialog, enter the file name of the artifact you want to customize, and click the green **Go** icon.

   When the file is located, it is displayed in the Application Resources panel.

5. Right-click the artifact, and choose **Customize**, and choose to add the associated library to the project.

   The artifacts from the imported library now display in the Application Navigator pane, under the **ADF Library Customizations** node, and the artifact you selected to customize opens in the editor window, as shown in Figure 10–6.

Figure 10–6 JSPX Page Open in Editor and Ready to Customize



Note: If imported data controls are not displaying in the Data Controls panel, do the following:

1. In the JDeveloper menu, go to **Tools > Preferences** to open the Preferences dialog.
2. Expand the **Business Components** node.
3. Select **General**.
4. Select **Display Imported ADF Libraries in Data Control Palette**.

---

6. In the Customization Context window (by default, displayed at the bottom of JDeveloper), select the layer that you want the customizations written to.

   All customizations for ADF business components must be done in the Global layer. View layer customizations can be made in any other layer except User. For more information about customization layers, see Section 1.2, "Understanding Customization Layers."

You are now ready to begin customizing your artifact. For more information about customizing specific artifacts, see the following chapters:

- Chapter 11, "Customizing and Extending ADF Application Artifacts"
- Chapter 14, "Customizing and Extending Oracle Enterprise Scheduler Jobs"
- Chapter 15, "Customizing Security for ADF Application Artifacts"
- Chapter 17, "Configuring End User Personalization"

- Chapter 18, "Customizing Help"

10.2.4 Importing Customizations into Your Workspace

There may be occasions when you need to import other customizations into your workspace. For example, someone else may have made customizations to an application module to which you need to make changes as well. Before you make your customization, you need to import that application module into your customization workspace.

If you need to import customizations made to a single page or page fragment, you can use the Manage Customizations dialog to download the file, as described in Section 2.4.1, "Downloading and Uploading Customization Files Using the Manage Customizations Dialog." Save the customization file(s) to a zip or JAR file.

If you need to import multiple customizations available in the metadata repository for an application, you use the `exportMetadata` Oracle WebLogic Scripting Tool command. For more information, see the "Application Metadata Management Commands" section of *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*. This command saves the customization files in a JAR file that you can import into your workspace. For procedures, see the "Viewing ADF Library Runtime Customizations from exported JARs" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

> **Tip:** You can also use Oracle Enterprise Manager Fusion Applications Control to import and export customization files. For more information, see the "Transferring Metadata Using Fusion Middleware Control" section of the *Oracle Fusion Middleware Administrator's Guide*. The referenced procedure describes using Fusion Middleware Control, but also applies to Fusion Applications Control.

If you want to use extensions (for example, if you want to add a custom entity object to an existing application module), the extensions must be deployed into the environment to which you have a connection. For more information, see Section 11.13, "Deploying ADF Customizations and Extensions."

10.2.5 Resynchronizing Your Customization Workspace Configuration Files

During the process of customization, it is possible that the base application that you are customizing will be updated with a patch. If this happens, you might need to resynchronize the configuration files in your local customization workspace from the exploded EAR of the application you are customizing.

When you create a customization workspace in your local development environment, workspace configuration files (such as, `adf-config.xml`, `connections.xml`, and `web.xml`) are copied to the local development environment. In some cases, the file is modified to allow you to implement and test customizations locally. When a patch is applied to the base application, these configuration files might change, and would therefore need to be synchronized to your local development environment so that you can continue to implement and test customizations.

JDeveloper allows you to check for and process updates to the customization workspace configuration files after a patch has occurred on the base application. When you run the check, there are three potential results for each file:

- The file in the local development environment does not need to be updated.

- The file in the local development environment needs to be updated, and can be updated safely because the local version has not been modified.
- The file in the local development environment needs to be updated, but cannot be updated safely because the local version has been modified.

After the check, JDeveloper lets you decide how to handle the update. If you choose to proceed with the updates, backups of the local files are created. You can use the backup files to manually merge changes into the updated files if necessary.

To synchronize your customization workspace configuration files:

1. Start JDeveloper in the **Oracle Fusion Applications Administrator Customization** role, and open your application customization workspace.
2. From the main menu, choose **Application** > **Synchronize Patch Changes**.

   The check is run, and the Synchronize Patch Changes dialog displays the results.
3. If no files in the development environment need updating, the Synchronize Patch Changes dialog gives you the option to review the list of possible updates. Click **Yes** to view possible updates, or **No** to close the dialog.
4. If one or more files need to be updated, the Synchronize Patch Changes dialog displays the files that might be out of date. Files that have been modified locally are indicated with a green icon. Click **Yes** to update the files, or **No** to skip the updates and close the dialog.

Note: If you choose to proceed with the updates, backups of the local files are created. You can use the backup files to manually merge changes into the updated files if necessary.

10.3 Customizing SOA Composites with JDeveloper

Before you begin customizing, you must identify the SAR file to customize, retrieve the configuration plan from the default composite in the MDS Repository, and set up the workspace and composite project for MDS Repository customization in JDeveloper using the **Oracle Fusion Applications Developer** role. After the workspace is created, you must exit and reenter JDeveloper using the **Oracle Fusion Applications Administrator Customization** role.

10.3.1 Before You Begin Using JDeveloper to Customize

Perform the following tasks before you begin customizing SOA composites with JDeveloper:

1. Identify the SAR file to customize, and locate it in the *APPLICATIONS_BASE*/fusionapps/applications/*product_family*/deploy directory. This directory includes the following files:
   - Composite SAR (sca_*.jar)
   - BPM template (bta_*.jar)
   - List resource bundle classes (jar_*.jar)
2. Ensure that the SAR file is marked as customizable by Oracle Fusion Applications. Otherwise, customizations do not survive patching or are not properly patched and merged. For information about which SOA composites are customizable, see

the product-specific documentation from Oracle Enterprise Repository for Oracle Fusion Applications.

If you encounter the following message when importing the SAR for customization, it means that Oracle Fusion Applications did not mark the composite for customizations in JDeveloper and your changes cannot survive patching.

```
The composite from the archive was not created for
customization. If you import the composite for
customization, you can customize it but you will have
problems to merge your customizations to any new
versions of that composite.
Do you want to continue?
Otherwise, uncheck "Import for Customization" box, and
click "Finish" option.
```

3. Get the configuration plan from the default composite in the MDS Repository using the following Oracle WebLogic Scripting Tool commands:

   a. Identify the default revision of the composite with `sca_getDefaultCompositeRevision`. For example:

   ```
   wls:/mydomain/ServerConfig> sca_getDefaultCompositeRevision("myhost",
   "7001", "weblogic", "weblogic",
   "FinGlCurrencyUserPreferredCurrencyComposite")
   ```

   b. Export the full composite corresponding to the default revision with `sca_exportComposite`. For example:

   ```
   wls:/offline/mydomain/ServerConfig> sca_
   exportComposite('http://myhost:8001', 'none', '/tmp/sca_
   FinGlCurrencyUserPreferredCurrencyComposite.0.jar',
   'FinGlCurrencyUserPreferredCurrencyComposite',
    '1.0')
   ```

   c. Extract the configuration plan used originally with the export action with `sca_extractPlan`. For example:

   ```
   wls:/mydomain/ServerConfig> sca_extractPlan("/tmp/sca_
   FinGlCurrencyUserPreferredCurrencyComposite_
   rev1.0.jar", "/tmp/FinGlCurrencyUserPreferredCurrencyComposite_
   configPlan.xml")
   ```

For information about using these commands, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

10.3.2 Setting Up the JDeveloper Workspace and Composite Project for MDS Repository Customization

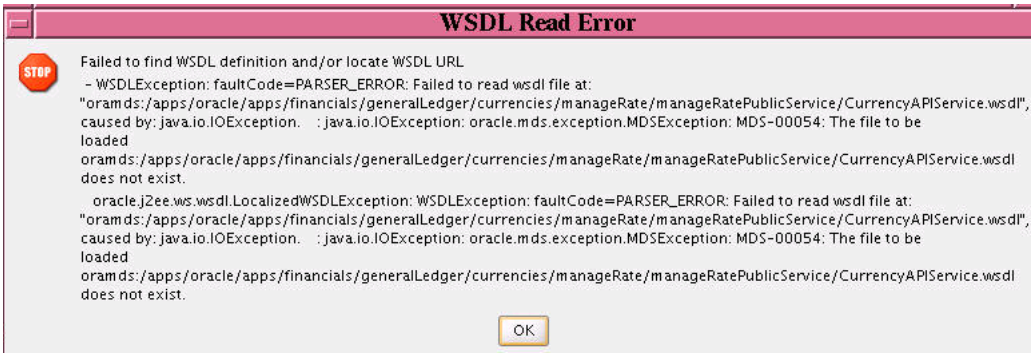
To set up the JDeveloper workspace and composite project for MDS Repository customization:

1. Start JDeveloper using the **Oracle Fusion Applications Developer** role.
2. From the **File** main menu, select **New** > **Applications** > **SOA Application** > **OK** to create a SOA application with an `XX` prefix in the application name.

   The `XX` prefix identifies an artifact or object created by the customer and distinguishes it from artifacts of Oracle Fusion Applications. You can delete the SOA project named **Project1** that was created by default.

The Oracle Fusion Applications composite references shared artifacts through the SOA-shared repository stored in the MDS Repository instead of replicating the shared artifact throughout the Oracle Fusion Applications code source. If the references to the SOA shared repository are not resolved, you receive the error message shown in Figure 10–7.

Figure 10–7 WSDL Read Error Message



3. To resolve references to the SOA-shared repository (`oramds:/apps`), define an MDS Repository entry in the `adf-config.xml` file to point to the SOA Infrastructure MDS Repository partition in the MDS Repository schema corresponding to the SOA cluster where you plan to deploy the customized composite. Add a `<namespace>` attribute with `path="/apps"` to `<metadata-namespaces>`:

```
<namespace metadata-store-usage="mstore-usage_2" path="/apps"/>
```

4. Add a `<metadata-store-usage>` attribute to `<metadata-store-usages>` for a database-based MDS Repository that points to the SOA Infrastructure MDS Repository partition in the SOA MDS Repository schema.

5. Replace the database schema name, database server, database port, and database name with actual values. To identify the user name, password, and database connection information, see the configuration for the MDS-SOA data source in the Oracle WebLogic Server Administration Console.

```
<metadata-store-usage id="mstore-usage_2">
   <metadata-store class-name="oracle.mds.persistence.stores.db.
    DBMetadataStore">
      <property value="soa_mds_schema_name" name="jdbc-userid"/>
      <property value="soa_mds_schema_password" name="jdbc-password"/>
      <property value="jdbc:oracle:thin:@database_server:
        database_port:database_name" name="jdbc-url"/>
      <property value="soa-infra" name="partition-name"/>
   </metadata-store>
</metadata-store-usage>
```

The following code shows an `<adf-mds-config>` example in `adf-config.xml`. The `mstore-usage_2` entry resolves references to the SOA shared repository:

```
<adf-mds-config xmlns="http://xmlns.oracle.com/adf/mds/config">
  <mds-config xmlns="http://xmlns.oracle.com/mds/config">
    <persistence-config>
      <metadata-namespaces>
        <namespace metadata-store-usage="mstore-usage_1" path="/soa/shared"/>
        <namespace metadata-store-usage="mstore-usage_2" path="/apps"/>
```

```
      </metadata-namespaces>
      <metadata-store-usages>
        <metadata-store-usage id="mstore-usage_1">
          <metadata-store
 class-name="oracle.mds.persistence.stores.file.FileMetadataStore">
            <property value="$\{oracle.home}/integration"
                      name="metadata-path"/>
            <property value="seed" name="partition-name"/>
          </metadata-store>
        </metadata-store-usage>
        <metadata-store-usage id="mstore-usage_2">
          <metadata-store
 class-name="oracle.mds.persistence.stores.db.DBMetadataStore">
            <property value="FIN_FUSION_MDS_SOA" name="jdbc-userid"/>
            <property value="FIN_FUSION_MDS_SOA" name="jdbc-password"/>
            <property
 value="jdbc:oracle:thin:@database_server.us.example.com:1521:database_name"
                      name="jdbc-url"/>
            <property value="soa-infra" name="partition-name"/>
          </metadata-store>
        </metadata-store-usage>
      </metadata-store-usages>
    </persistence-config>
  </mds-config>
</adf-mds-config>
```

6. From the **File** main menu, select **Import** > **SOA Archive Into SOA Project** to import the SAR, then click **OK**.
7. In the **Project Name** field, enter the name of the new SOA project with an `XX` prefix and select a name to identify the base composite that you are extending. For example, specify `XXFinGlCurrencyUserPreferredCurrencyComposite` if you are customizing `FinGlCurrencyUserPreferredCurrencyComposite`.
8. Click **Next**.
9. In the **Composite Archive** field, perform the following steps:
   a. Click **Browse** to select the SAR to customize that you identified in Section 10.3.1, "Before You Begin Using JDeveloper to Customize."
   b. Accept the default setting for the composite name.
   c. Select the **Import for Customization** checkbox.
   d. Click **Finish**.

   You accept the default composite name to ensure that patching and SOA can identify whether runtime customizations, JDeveloper customizations, or both types have been applied to the composite. If the composite is renamed, patching and SOA have no knowledge of the renamed composite.

   You may see an error icon on a partner link in the `composite.xml` design view that reports the following error:

   ```
   Couldn't resolve classpath:/META-INF/wsdl/ServiceException.wsdl
   ```

   This error is addressed in subsequent steps.
10. Right-click the composite project and go to **Project Properties** > **Libraries and Classpath**.
11. Click **Add Library**, and select the **BC4J Service Client** library.

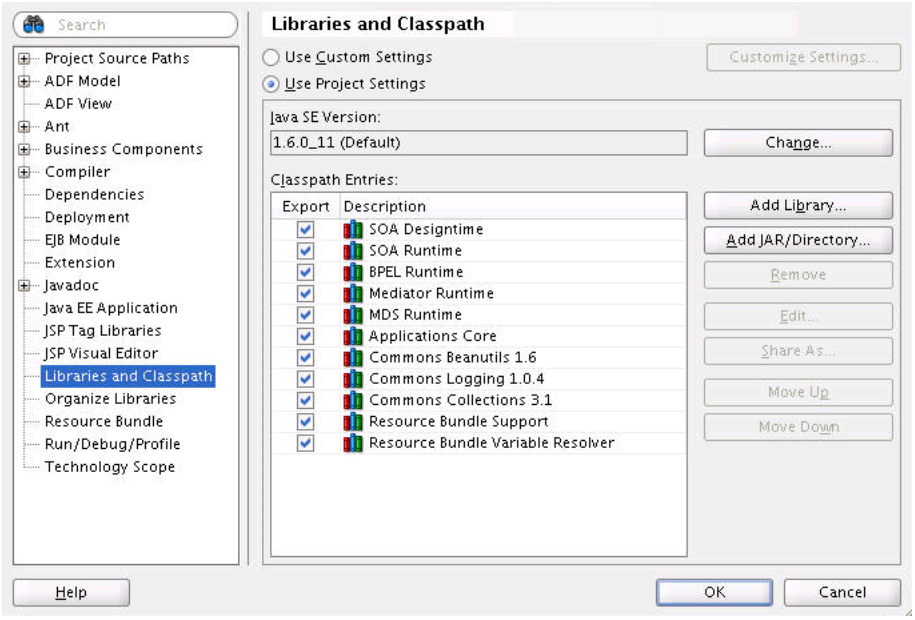
12. Click **OK** to close the Add Library dialog.
13. Click **OK** to close the Project Properties dialog.

    By adding this library to your SOA project, you avoid the design time error you may have received in Step 9d.
14. Click the **Validate** icon in the `composite.xml` design view. The error shown in Step 9d that you may have received for the partner link should now be resolved.
15. Make the customization classes and values available in your project.

    There are two types of customization classes:

    - Applications Core customizing classes are available from the Application Core shared library. See Section 1.2, "Understanding Customization Layers" for the list of Applications Core customization classes permitted in JDeveloper.
    - Product team customization classes are available in the appropriate EAR file. These customization classes are bundled in a JAR file in the EAR's `APP-INF/lib` directory. These JAR files follow a naming convention of `Ext...jar`. Therefore, you must get these JAR files from the deployed area, and perform the following steps:
      - Put the customization class JAR file under `$JDEV_HOME/jdev/extensions`.
      - Add the JAR file in the new project's library and class path setting.
16. Right-click the composite project and go to **Project Properties** > **Libraries and Classpath**.
17. Add the **Applications Core** library to the composite project, as shown in Figure 10–8.

Figure 10–8 Applications Core Library



18. Go to **Application Resources** > **Descriptors** > **ADF META-INF** > **adf-config.xml**.
19. Add the appropriate customization class in the MDS Repository configuration, such as `oracle.apps.fnd.applcore.customization.SiteCC`.

20. Right-click the composite project and go to **Project Properties** > **Libraries and Classpath**.

    The following libraries have now been added:

    - Application Core
    - BC4J Service Client

10.3.3 Customizing the Composite

To customize the composite:

1. Start JDeveloper with the **Oracle Fusion Applications Administrator Customization** role.
2. Select the value for the layer in the Customization Context dialog that you want to customize. Figure 10–9 provides details.

Figure 10–9 Customization Context Dialog

3. See Chapter 12, "Customizing and Extending SOA Components" for instructions on customizing the composite during design time in JDeveloper and runtime with Oracle SOA Composer, Oracle BPM Worklist, and Oracle Enterprise Manager Fusion Applications Control.
4. When introducing new components, partner links, and artifacts to the composite, prefix the names with XX to prevent problems with existing and future components that may be introduced in Oracle Fusion Applications patches.
5. Use the configuration plan that you extracted in Step 3 of Section 10.3.1, "Before You Begin Using JDeveloper to Customize." If any new partner links were added to your composite, add entries to the configuration plan, if needed. For information about configuration plans, see the "Customizing Your Application for the Target Environment Prior to Deployment" section of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
6. Deploy the composite using the same revision you found in Section 10.3.1, "Before You Begin Using JDeveloper to Customize."

10.3.4 Customizing SOA Resource Bundles

Table 10–1 describes how to customize SOA resource bundles.

Table 10–1 Customizing SOA Resource Bundles

To Modify or Add Translatable Strings For...	Description
An existing human task, activity guide, or BPEL process	This feature is not supported in the first version of Oracle Fusion Applications (for example, modifying the human task title). Runtime modifications do not support this functionality and the `.task`, `.ag`, and `.agdl` files are not customizable in JDeveloper.
Human task mapped attributes	This feature is not supported in the first version of Oracle Fusion Applications. Translations for human task mapped attribute labels are defined in the following resource bundle: `oracle.bpel.services.workflow.resource.WorkflowLabels` For this version, the `WorkflowLabels` resource bundle is deployed to the SOA clusters in the Customer Relationship Management (CRM) and Human Capital Management (HCM) domains. Any customizations to the resource bundle are overwritten with future patches.
Server name in Federated Worklist on the Oracle Fusion Applications home page	The server names that appear in the Federated Worklist on the Oracle Fusion Applications home page are defined in the following file: `oracle/apps/common/acr/resource/ResourcesAttrBundle.xliff` See Section 11.12, "Customizing or Adding Resource Bundles" for instructions on overriding strings in XLIFF resource bundles.

Note: Oracle Fusion Applications automatically seed human task-protected mapped attributes and labels, but do not seed public mapped attributes. If you require human task mapped attributes, it is recommended that you use the public mapped attributes. However, if protected mapped attributes are required, then prefix your label names with `XX` to prevent problems with Oracle Fusion Applications seeded labels.

11

Customizing and Extending ADF Application Artifacts

This chapter describes how to use Oracle JDeveloper to customize and extend application artifacts defined by Oracle Application Development Framework (Oracle ADF) in Oracle Fusion applications.

This chapter includes the following sections:

11.1 About Customizing Oracle ADF Application Artifacts

With the customization features provided by Oracle Metadata Services (MDS), developers can customize Oracle Fusion Applications using JDeveloper, making modifications to suit the needs of a particular group, such as a specific country or site.

Using JDeveloper, you can implement customizations on existing artifacts that are stored in a metadata repository and retrieved at run time to reveal the customized application. You can also extend you application with new custom artifacts that are packaged into a JAR file, and integrated using customizations on the existing application.

Note that many kinds of customization can be performed in the runtime environment using CRM Application Composer, which allows you to customize existing objects and extend an application with new objects for the following CRM applications:

- Sales
- Marketing
- Customer Center
- Trading Community Architecture
- Order Capture

For more information about using CRM Application Composer to customize these applications, see Chapter 4, "Customizing Objects."

However some kinds of customization (including all customizations to applications other than those listed above) require a lower level approach, for which you will need to use JDeveloper.

11.1.1 Before You Begin Customizing Oracle ADF Application Artifacts

Before you customize application artifacts (such as entity objects, view objects, application modules, and pages) using JDeveloper, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

Before you make any changes to the data model such as adding entity objects or attributes, first check to see if there are existing flexfields that meet your needs. For more information, see Chapter 5, "Using Flexfields for Custom Attributes."

In addition, you will need to perform the following tasks before you can begin customizing your application:

- Set up a test environment.

  All application artifact customizations should be deployed to a test environment. For more information, see Chapter 2, "Understanding the Customization Development Lifecycle."

- Determine which artifacts you want to customize.

  Before you can implement customizations using JDeveloper, you must first determine which business objects you want to customize, so that you can create a workspace that imports the necessary parts of the application. For more information, see Section 10.2, "Customizing Oracle ADF Artifacts with JDeveloper."

- Create an application workspace.

  Before you can implement customizations using JDeveloper, you must create a workspace that imports the necessary parts of the application you want to customize. For more information, see Section 10.2.1, "Creating the Customization Application Workspace."

- Launch JDeveloper in the appropriate role.

  If you are implementing customizations on existing application artifacts, you must select the **Oracle Fusion Applications Administrator Customization** role when you launch JDeveloper.

If you are creating new custom application artifacts (such as, entity objects, view objects, and pages), you must select the **Oracle Fusion Applications Developer** role when you launch JDeveloper.

- Select a layer value.

  When customizing application artifacts in JDeveloper, you first need to select the layer and layer value to work in. You use the Customization Context window to make this selection. When customizing business components, such as entity objects and view objects, you must use the `global` layer. For more information about customization layers, see Section 1.2, "Understanding Customization Layers."

11.2 Editing Existing Business Components

When customizing an application in JDeveloper, be aware that the layer in which you choose to implement customizations has an impact on what kinds of customizations you can perform. If you want to customize an ADF Business Components object, such as an entity object or view object, you must use the `global` layer.

Before You Begin

Before you start customizing business objects, you'll need to determine which business objects you want to customize and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

Then when customizing ADF artifacts, you need to launch JDeveloper in the **Oracle Fusion Applications Administrator Customization** role, and select the `global` layer.

Task: Edit Attributes

You can customize the properties of an attribute from an entity object or view object using JDeveloper. When you open an entity object or view object in the overview editor, you click the Attributes tab to see the attributes of the object. When you select an attribute, its properties are displayed in the Property Inspector.

It is not necessary to modify the page after customizing the properties of an existing attribute. Customizations to existing attributes are automatically reflected on the pages that show them.

However, if you modify an attribute so that it requires a different UI component, you must also update the page. For example, if you add a list of values (LOV) to an attribute, you will need to edit the page to hide the existing UI component that displays the attribute, and add a new UI component that can display the LOV.

Note that some attribute properties defined in the entity object can be overridden in the view object. For example, you can define the label text for a field in an entity object and subsequently give it a different label in the consuming view object. Then pages that use the view object display the label from the view object.

For more information about attributes in entity objects, see the "Creating a Business Domain Layer Using Entity Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Add Attributes

You can add custom attributes to an entity object or view object using JDeveloper. To do this, you must launch JDeveloper in the **Oracle Fusion Applications Administrator Customization** role, and select the global layer. When you open an entity object or

view object in the overview editor, you click the Attributes tab to see the attributes of the object. To add a custom attribute, click the **Add** icon.

If you want your custom attribute to be stored in the database, you must first create the column that will be use to store it.

If you want your custom attributes to be displayed in the application, you must also customize the pages to display them. For more information, see Section 11.4, "Editing Pages."

For more information about attributes in entity objects, see the "Creating a Business Domain Layer Using Entity Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Edit Entity Objects

In JDeveloper, you edit entity objects using the overview editor. In the Application Navigator, right-click an entity object, and choose **Open**. Then click on the navigator tabs to view and edit the various features of the entity object.

For more information about entity objects, see the "Creating a Business Domain Layer Using Entity Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more information about using entity objects in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Edit View Objects

In JDeveloper, you edit view objects using the overview editor. In the Application Navigator, right-click a view object, and choose **Open**. Then click on the navigator tabs to view and edit the various features of the view object.

For more information about view objects, see the "Defining SQL Queries Using View Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more information about using view objects in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Edit Validation

In JDeveloper, you edit declarative validation rules for entity objects and view objects using the overview editor. In the Application Navigator, right-click an entity object or view object, and choose **Open**. Then click the Business Rules navigator tab to view and edit the validation rules.

When implementing customizations on validation rules, you can add rules, modify the error message, and make rules more restrictive. But you should avoid removing rules or making rules less restrictive, because this can cause unpredictable results. Also, you can edit only declarative validation rules; programmatic validation rules cannot be customized.

For more information, see the "Defining Validation and Business Rules Declaratively" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Edit Application Modules

In JDeveloper, you edit application modules using the overview editor. In the Application Navigator, right-click an application module, and choose **Open**.

In JDeveloper, you can make the following kinds of customizations on an application module:

- Add new custom properties. This is done on the General page of the overview editor.
- Add new view object and application module instances. This is done on the Data Model page of the overview editor.
- Add newly created subtype view objects. This is done on the Data Model page of the overview editor.
- Add new application module configurations. This is done on the Configurations page of the overview editor.

It is important to note that you cannot modify the web service interface for a service-enabled application module. You can, for example, add an attribute in a view object that is included in a service-enabled application module, but that attribute cannot be added to the web service interface.

For more information about working with application modules, see the "Implementing Business Services with Application Modules" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Add Customizations to Existing Reports

After you have implemented customizations on your application, you can use Oracle Business Intelligence Publisher to include these customizations in your reports. For more information, see Chapter 8, "Customizing Reports and Analytics."

11.3 Editing Task Flows

You can use JDeveloper to implement customizations on the task flows that are used in your application. A task flow is a set of ADF Controller activities, control flow rules, and managed beans that interact to allow a user to complete a task. Although conceptually similar, a task flow is not the same as a human task, a task in the worklist, or a process flow.

A bounded task flow can be rendered in a JSF page or page fragment (`.jsff`) by using an ADF region. This is typically done to allow reuse of the task flow, as necessary, throughout the application. If you modify a bounded task flow, the changes apply to any ADF region that uses the task flow. For more information, see the "Using Task Flows as Regions" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Before You Begin

Before you start editing task flows, you'll need to determine which task flows you want to customize, and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

When editing a task flow in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Administrator Customization** role.

Task: Edit Task Flows

In JDeveloper, you use the task flow diagram editor to implement customizations on existing task flows. In the Application Navigator, right-click the task flow you want to customize, and choose **Open**. The page is displayed in the diagram editor, where you can make changes to the existing activities and control flow cases, or create new custom ones. For more information, see the "Getting Started with ADF Task Flows" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

11.4 Editing Pages

You can use JDeveloper to implement customizations on the pages that are used in your application. When editing a page in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Administrator Customization** role.

Before You Begin

Before you start editing pages, you'll need to determine which pages you want to customize, and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

Task: Edit Pages

In JDeveloper, you use the visual editor to implement customizations on existing pages. In the Application Navigator, right-click the page you want to customize, and choose **Open**. The page is displayed in the visual editor (accessed by clicking the **Design** tab). Then you can edit the page as you typically would using this editor. For more information about editing pages in JDeveloper, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

11.5 Creating Custom Business Components

You can use JDeveloper to extend your application by creating custom business components. When creating custom business components in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Developer** role. This role is used for creating new custom objects that you want to add to the application. You can use the same workspace that you created for customization. Then after you have created the custom business components, you switch to the **Oracle Fusion Applications Administrator Customization** role, to make changes to existing artifacts to integrate the new custom artifacts into the application.

Before You Begin

Before you start creating business objects, you'll need to determine which business objects you want to create, and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

Task: Create Custom Entity Objects

An entity object represents a row in a database table, and encapsulates the business logic and database storage details of your business entities.

In JDeveloper, you can create entity objects using the Create Entity Object wizard, which you can launch from the New Gallery. In the Application Navigator, right-click the project you want to add the entity object to, and choose **New**. Then in the New Gallery, expand **Business Tier**, click **ADF Business Components**, choose **Entity Object**, and click **OK**. Follow the prompts in the wizard to create an entity object.

For more information about creating entity objects, see the "Creating a Business Domain Layer Using Entity Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more information about using entity objects and view objects in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Create Custom View Objects

A view object represents a SQL query and also collaborates with entity objects to consistently validate and save the changes when end users modify data in the UI.

In JDeveloper, you can create view objects using the Create View Object wizard, which you can launch from the New Gallery. In the Application Navigator, right-click the project you want to add the view object to, and choose **New**. Then in the New Gallery, expand **Business Tier**, click **ADF Business Components**, choose **View Object**, and click **OK**. Follow the prompts in the wizard to create a view object.

For more information about creating view objects, see the "Defining SQL Queries Using View Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more information about using entity objects and view objects in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Create Custom Application Modules

An application module encapsulates an active data model and the business functions for a logical unit of work related to an end-user task.

In JDeveloper, you can create application modules using the Create Application Module wizard, which you can launch from the New Gallery. In the Application Navigator, right-click the project you want to add the application module to, and choose **New**. Then in the New Gallery, expand **Business Tier**, click **ADF Business Components**, choose **Application Module**, and click **OK**. Follow the prompts in the wizard to create an application module.

For more information about creating application modules, see the "Implementing Business Services with Application Modules" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more information about using application modules in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Create a Web Service Interface for a Custom Application Module

In JDeveloper, you can edit a custom application module to create a web service interface that exposes the top-level view objects and defines the available service operations it supports. To do this, open the application module in the overview editor, click the **Service Interface** navigation tab, and click the **Enable support for Service Interface** icon. Then use the Create Service Interface wizard to configure the desired options.

It is important to note that the new web service cannot be deployed to the Fusion application. You can deploy it only to a new application.

For more information about creating a web service interface for an application module, see the "Integrating Service-Enabled Application Modules" chapter in the

Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.

For more information about using application modules in Fusion applications, see the "Getting Started with Business Services" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Add Validation

In JDeveloper, you can create declarative validation rules for entity objects and view objects to help ensure the integrity of the data. To do this, open the entity object or view object in the overview editor, and click the **Business Rules** navigation tab. Then select the attribute you want to provide validation for, click the **Create new validator** icon, and use the Add Validation Rule dialog to configure the rule. For more information, see the "Defining Validation and Business Rules Declaratively" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Enforce Data Security for a Custom Business Object

You can use JDeveloper to enforce row and attribute security for custom ADF Business Components objects. This is done using data security policies to secure data from business objects based on the grants made to roles.

When you need to expose data records in an extended application, you can use JDeveloper to create entity objects based on secured database resources, and then opt into data security policies by enabling row-level privilege checking for specific operations on the entity objects. For more information, see Section 15.5, "Enforcing Data Security in the Data Model Project."

Task: Add a Business Object to an Existing Report

After you have extended your application with custom business objects, you can use Oracle Business Intelligence Publisher to include these extensions in your reports. For more information, see Chapter 8, "Customizing Reports and Analytics."

11.6 Creating Custom Task Flows

You can use JDeveloper to create custom task flows that you can include in your application. A task flow is a set of ADF Controller activities, control flow rules, and managed beans that interact to allow a user to complete a task. Although conceptually similar, a task flow is not the same as a human task, a task in the worklist, or a process flow.

Before You Begin

Before you start creating custom task flows, you'll need to determine which task flows you want to create, and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

When extending your application with custom task flows in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Developer** role.

Task: Create a Custom Task Flow

You can create a custom task flow in JDeveloper using the New Gallery, and then define its activities using the task flow diagram editor. In the Application Navigator, right-click the project you want to add the task flow to, and choose **New**. Then in the New Gallery, expand **Web Tier**, and click **JSF/Facelets**. Then select **ADF Task Flow**,

and click **OK**. In the Create Task Flow dialog, you'll specify the details about the type of task flow you want to create. When you click **OK**, the task flow is created and displayed in the diagram editor.

For information about creating and designing task flows, see the "Getting Started with ADF Task Flows" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

11.7 Creating Custom Pages

You can use JDeveloper to create custom pages that you can include in your application. When creating custom pages in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Developer** role.

When creating the page (or dropping a view activity onto a task flow), you can create the page either as a JSF JSP or as a JSF JSP fragment. JSF fragments provide a simple way to create reusable page content in a project, and are what you use when you want to use task flows as regions on a page. When you modify a JSF page fragment, the JSF pages that consume the page fragment are automatically updated.

After extending your application with custom pages, you will need to make sure that security for the new pages is implemented appropriately and that the new pages are deployed so that they are accessible from the application. For more information about updating security, see Chapter 15, "Customizing Security for ADF Application Artifacts." For more information about deployment, see Section 11.13, "Deploying ADF Customizations and Extensions."

For more information about creating pages in JDeveloper, see the following resources:

- The *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*
- "Getting Started with Your Web Interface" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- "Implementing UIs in JDeveloper with Application Tables, Trees and Tree Tables" in the *Oracle Fusion Applications Developer's Guide*
- "Implementing Applications Panels, Master-Detail, Hover, and Dialog Details" in the *Oracle Fusion Applications Developer's Guide*
- "Creating Customizable Applications" in the *Oracle Fusion Applications Developer's Guide*

Before You Begin

Before you start creating custom pages, you'll need to determine which pages you want to create, and create an application customization workspace. For more information, see Section 11.1.1, "Before You Begin Customizing Oracle ADF Application Artifacts."

When creating custom pages in JDeveloper, you must launch JDeveloper in the **Oracle Fusion Applications Developer** role.

Task: Create a Custom Page

In JDeveloper, you can create pages either by double-clicking a view activity in a task flow or by using the New Gallery. In the Application Navigator, right-click the project you want to add the page to, and choose **New**. Then in the New Gallery, expand **Web Tier**, and click **JSF/Facelets**. Then select either **Page** or **ADF Page Fragment**, and click **OK**.

Task: Add a Custom Page to a Task Flow

If you created the page by double-clicking a view activity in a task flow, it is already added to the task flow. If you created it using the New Gallery, you can add it to a task flow by dragging the page from the Application Navigator and dropping it in the task flow diagram editor. Then you can connect the page using a control flow. For more information, see the "Getting Started with ADF Task Flows" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Enable Runtime Customization for a Custom Page

To enable a custom page to be customized at runtime, you must make sure that the page and the project that contains it are set to allow runtime customizations. For information on how to do this, see the "Authorizing Runtime Customization of Pages and Task Flows" section in the *Oracle Fusion Applications Developer's Guide*.

11.8 Customizing and Extending the Oracle Fusion Applications Schemas

Using the database tools of your choice, you can customize and extend the Oracle Fusion Applications schemas to suit the needs of your organization. However, you should first consider using CRM Application Composer or flexfields to satisfy your additional data storage requirements. For more information about using CRM Application Composer, see Chapter 4, "Customizing Objects." For more information about using flexfields, see Chapter 5, "Using Flexfields for Custom Attributes."

11.8.1 About Customizing and Extending the Oracle Fusion Applications Schemas

If you need to extend the preconfigured Oracle Fusion Applications schemas to address additional data storage needs, you should create a custom schema. In your custom schema, you can create tables, columns, and all the necessary additional schema objects that you need. This approach allows you to contain and maintain all of your custom data storage objects separately from the preconfigured Oracle Fusion Applications schemas.

If necessary, you can extend the preconfigured schemas within certain constraints. With the exception of customizing a preconfigured table to include new custom objects, such as columns, you must not make any customizations to preconfigured schema objects. Instead, you can extend the schema by adding new custom objects that you can configure as needed.

When making amendments to the schema, such as adding tables or columns, prefix the name of the table or column with a unique identifier (for example, `XX_`) to prevent collisions with existing objects.

Any code that accesses the new custom schema objects should use fully qualified table names (for example, *`SCHEMA_NAME.TABLE_NAME`*). If your code does not use fully qualified table names, you will need to create synonyms for the custom tables. The synonym must be created in the FUSION schema, and associated privileges must be granted in the FUSION_RUNTIME schema. At runtime, Oracle Fusion applications connect to the FUSION_RUNTIME schema, so privileges must be granted there. However, because the schema context is set to FUSION, the synonym must be created there. This convention applies in all cases, whether you create custom schema objects in a custom schema or a preconfigured schema.

For information about creating database objects, see the Designing Databases topics in the JDeveloper online help.

11.8.2 What You Can Do With Schema Modifications

Using the SQL Worksheet in JDeveloper or the database tools of your choice, you can issue commands to the database to customize and extend it. When making changes to the database, you can do the following:

- Add a custom schema
- Add or modify tables
- Add columns to preconfigured or custom tables
- Add indexes to custom columns
- Add sequences
- Add PL/SQL packages, procedures, functions and abstract data types

11.8.3 What You Cannot Do With Schema Modifications

When making changes to the database, you cannot do any of the following:

- Modify preconfigured columns or sequences.
- Modify preconfigured PL/SQL packages, procedures, functions and abstract data types (unless explicitly directed to do so by product documentation).
- Delete preconfigured schema objects.
- Add indexes to preconfigured columns (unless explicitly directed to do so by product documentation).

11.8.4 Before You Begin Extending the Oracle Fusion Applications Schemas

Before you modify the Oracle Fusion Applications schema, you should first see if you can address your additional data storage requirements using flexfields, as described in Chapter 5, "Using Flexfields for Custom Attributes."

11.8.5 Extending the Schemas Using a Custom Schema

Using the SQL Worksheet in JDeveloper, you can issue commands to the database to customize and extend it. In a custom schema, you can add tables, columns, indexes, and other schema objects to support the customizations and extensions you want to implement in the application (such as, adding an attribute to an entity object).

To access the SQL Worksheet, right-click the database connection (under the **Connections** node in the Application Resources panel of the Application Navigator), and choose **Database Navigator** from the context menu. This will open the selected database connection in the Database Navigator and display the SQL Worksheet.

Before You Begin

Before you attempt to extend the schema, you should be familiar with the guidelines described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

Task: Create a Custom Schema

When making creating a custom schema, prefix the name of the schema with a unique identifier (for example, `XX_`) to prevent collisions with existing schemas. You must grant the privileges to the custom schema that are necessary for it to function properly and for any supporting code to compile (for example, objects referenced in PL/SQL code).

Task: Create Custom Database Tables, Columns, Indexes, and Sequences

Within a custom schema, you can create custom database tables, columns, indexes, and sequences to address your additional data storage needs. When adding custom objects, prefix the name of the object with a unique identifier (for example, `XX_`) to prevent collisions with existing objects. New custom indexes and sequences should adhere to this convention as well.

After creating a custom table, you will need to grant the necessary object privileges to the FUSION_RUNTIME schema, which Oracle Fusion Applications uses at run time. You can grant privileges directly to the schema, or through a custom database role, but do not use the preconfigured FUSION_APPS_READ_AND_WRITE database role.

Any code that accesses the new custom schema objects should use fully qualified table names (for example, *`SCHEMA_NAME.TABLE_NAME`*). If your code does not use fully qualified table names, you will need to create synonyms for the custom tables, as described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

Task: Create Custom PL/SQL Packages, Procedures, Functions, and Abstract Data Types

When adding PL/SQL objects and abstract data types to a custom schema, prefix the name of the object or data type with a unique identifier (for example, `XX_`) to prevent collisions with existing objects.

Your PL/SQL code should contain the `AUTHID INVOKER` clause so that the code is executed within the context of the privilege set of the FUSION_RUNTIME user. Additionally, the FUSION_RUNTIME user must be granted the EXECUTE privilege on the PL/SQL object or type, either directly or through a database role.

If you need to create synonyms to support your PL/SQL code, create your synonyms in the FUSION schema, as described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

11.8.6 Extending a Preconfigured Schema

Using the SQL Worksheet in JDeveloper, you can issue commands to the database to customize and extend it. When making changes to the schema, you can add tables or columns to support the customizations and extensions you want to implement in the application (such as, adding an attribute to an entity object). However, do not remove tables or columns, as this can have adverse affects in other parts of the application.

With the exception of customizing a preconfigured table to include new custom objects, such as columns, you must not make any customizations to preconfigured schema objects.

To access the SQL Worksheet, right-click the database connection (under the **Connections** node in the Application Resources panel of the Application Navigator), and choose **Database Navigator** from the context menu. This will open the selected database connection in the Database Navigator and display the SQL Worksheet.

Before You Begin

Before you implement extensions to a preconfigured schema, consider creating your extensions in a custom schema. This approach provides greater flexibility and modularity.

Also, you should be familiar with the guidelines described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

Task: Edit Database Tables

With the exception of customizing a preconfigured table to include new custom objects, such as columns, you must not make any customizations to preconfigured schema objects.

When adding columns to a preconfigured table, prefix the name of the column with a unique identifier (for example, `XX_`) to prevent collisions with existing columns.

Task: Create Custom Database Tables, Columns, Sequences, and Indexes

You can create custom database tables and columns to address your additional data storage needs. When adding custom tables and columns, prefix the name of the table and columns with a unique identifier (for example, `XX_`) to prevent collisions with existing tables and columns.

After creating a custom table, you will need to grant the necessary object privileges to the FUSION_RUNTIME schema, which Oracle Fusion Applications uses at run time. You can grant privileges directly to the schema, or through a custom database role, but do not use the preconfigured FUSION_APPS_READ_AND_WRITE database role.

Any code that accesses the new custom schema objects should use fully qualified table names (for example, *`SCHEMA_NAME.TABLE_NAME`*). If your code does not use fully qualified table names, you will need to create synonyms for the custom tables, as described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

You can create new custom indexes on custom columns, but do not attempt to create an index on a preconfigured column, unless explicitly directed to do so by product documentation.

Task: Create Custom PL/SQL Packages, Procedures, Functions, and Abstract Data Types

When adding PL/SQL objects and abstract data types, prefix the name of the object or data type with a unique identifier (for example, `XX_`) to prevent collisions with existing objects.

Your PL/SQL code should contain the `AUTHID INVOKER` clause so that the code is executed within the context of the privilege set of the FUSION_RUNTIME user. Additionally, the FUSION_RUNTIME user must be granted the EXECUTE privilege on the PL/SQL object or type, either directly or through a database role.

If you need to create synonyms to support your PL/SQL code, create your synonyms in the FUSION schema, as described in Section 11.8.1, "About Customizing and Extending the Oracle Fusion Applications Schemas."

11.9 Customizing or Creating a Custom Search Object

In JDeveloper, you can customize and create saved searches and search forms for your application. To customize a search form or saved search in JDeveloper, you'll need to set up an application workspace as described in Section 10.2.1, "Creating the Customization Application Workspace." Then, locate and open the object you want to customize. To create a new search form, you open or create the page that will display the form and select a data collection from the Data Controls panel. For more information, see the "Creating ADF Databound Search Forms" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

11.10 Editing the UI Shell Template

For CRM applications, you can use Page Composer to edit the UI shell template, as described in Section 3.4, "Editing the UI Shell Template Used by All Pages." For other Oracle Fusion Applications, you will need to use JDeveloper.

To edit the UI Shell template in JDeveloper, you'll need to set up an application workspace as described in Section 10.2.1, "Creating the Customization Application Workspace." Then, in the **Oracle Fusion Applications Administrator Customization** role, expand the contents of the **Applications Core (ViewController)** library and drill down to the file `oracle/apps/fnd/applcore/templates/UIShell.jspx`. This is the UI Shell template, which you can customize as necessary.

Alternatively, you can access the UI Shell template from any page in the library. Open the page in JDeveloper, right-click on the view ID of the `pageTemplate` tag (`/oracle/apps/fnd/applcore/templates/UIShell.jspx`), and then choose **Go to Declaration** to open the UI Shell template.

You can also customize the Oracle Fusion Applications skin (for both CRM and non-CRM applications) as described in Chapter 19, "Customizing the Oracle Fusion Applications Skin."

11.11 Customizing Menus

Using JDeveloper you can customize the menus in your Oracle Fusion applications. Customizing the tasklist menu follows the same pattern as editing most artifacts (such as, pages or business components) from the EAR connection. However, customizing the home page, preferences and navigator menus is slightly different. For these menus, you will need to export the menu's XML file from the MDS repository and copy them into your customization workspace before you can implement customizations.

Note: You can also customize the navigator menu at runtime from the Setup and Maintenance work area, as described in Chapter 6, "Customizing the Navigator Menu."

When exporting the menu XML files from the MDS repository, as described in Section 10.2.4, "Importing Customizations into Your Workspace," you can find them in the `oracle/apps/menu` directory in the repository. The following are their file names:

- home page menu: `homepage_menu.xml`
- preferences menu: `pref_menu.xml`
- navigator menu: `root_menu.xml`

Then you copy the files to the same directory (under project source path) in your local workspace (for example, *`CUSTOMIZATION_APP_PATH`*`/`*`PROJECT_NAME`*`/src/oracle/apps/menu`). After you have copied them into your local customization workspace, you can customize the menus as necessary.

After you have implemented customizations on a menu, you will need to update the MAR profile to make sure they are included during deployment. In the MAR profile, under **User Metadata** > **Directories**, select the customizations you implemented that correspond to the menu files. For more information about deploying customizations, see Section 11.13, "Deploying ADF Customizations and Extensions."

For more information about menus in Oracle Fusion Applications, see the "Working with the Global Menu Model" section in *Oracle Fusion Applications Developer's Guide*.

11.12 Customizing or Adding Resource Bundles

One method of customizing text is defining a new key in the resource bundle. There is a single override resource bundle for Oracle Fusion Applications. You can use this resource bundle to override values for existing keys, but you cannot add new keys.

Because you cannot define a new key in the shipped resource bundle, you need to create a new override bundle. You can accomplish this in JDeveloper by creating an XLIFF file from the New Gallery. After the file is generated, you can then enter new keys and their associated text in the XLIFF file.

To make the newly created resource bundle available for customization, you need to register the resource bundle with the customization project. You can do this from the Resource Bundle page of the Project Properties dialog.

You can also extend your application by creating a new resource bundle for a project if, for example, you want to customize the text for a label and you don't want to change the value in the global override bundle. To do this, create an XLIFF file from the New Gallery, package it into an ADF Library JAR file, and import the JAR file into the customization project.

Note: All custom JAR file names must begin with the prefix Xx, for example `XxMyJar.jar`

To test your customizations locally in JDeveloper Integrated WebLogic Server, you must also include the ADF Library JAR file in the `APP-INF/lib` directory.

For information about translating custom resource bundle strings, see Section 16.2, "Translating Resource Bundles from Metadata Services Metadata Repository."

For more information about working with resource bundles, see the "Creating a Business Domain Layer Using Entity Objects" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

11.13 Deploying ADF Customizations and Extensions

After you have customized existing artifacts, you can use JDeveloper to deploy the customizations to a sandbox or to the Fusion application. For more information on how to use sandboxes to isolate changes from the mainline so you can test and validate the changes, see Section 2.2.2, "Setting Up Sandboxes."

When you create a customization workspace as described in Section 10.2.1, "Creating the Customization Application Workspace," the wizard generates a MAR profile. By default, the name of the MAR profile is *`application_name`*`_customizations`. It will automatically include the customizations that you implement. You can use this profile to package your customizations for deployment.

When you package customizations from the customization workspace, the MAR file should include only library customizations. Do not include the **User Metadata** or **HTML Root Dir for Project** in the MAR profile, unless explicitly directed to do so by product documentation.

If you extend your application with new custom artifacts, you can use JDeveloper to package them into an ADF Library JAR and place them into the proper location within the application directory structure.

Task: Deploy the Customizations

You can use JDeveloper to deploy the customizations directly or you can use JDeveloper to create a MAR, and then load the MAR using WLST commands or the WebLogic Server Administration Console.

> **Tip:** You can also use Oracle Enterprise Manager Fusion Applications Control to import and export customization files. For more information, see the "Transferring Metadata Using Fusion Middleware Control" section of the *Oracle Fusion Middleware Administrator's Guide*. The referenced procedure describes using Fusion Middleware Control, but also applies to Fusion Applications Control.

If you are using JDeveloper to deploy directly, you have a choice to deploy to available sandboxes or into the already deployed Fusion application.

When you deploy customizations on ADF Business Component objects (such as entity objects and view objects), the server must be restarted for the customizations to be picked up.

For instructions on deploying customizations, see the section on "How to Deploy New Customizations Applied to ADF LIbrary" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Package New Artifacts into ADF Library

If you have extended your application with new custom artifacts (or you are supplied with new artifacts), you must package these artifacts into an ADF library JAR and place the JAR files in the proper location within the application.

> **Note:** All custom JAR files must begin with the prefix Xx, for example `XxMyJar.jar`

The ADF library JAR for the new model artifacts (such as entity objects and view objects) should be placed into the *`ExplodedEarDirectory`*`/APP-INF/lib` directory (for example, `/fusionapps/applications/fin/deploy/EarFinPayables.ear/APP-INF/lib/XxMyJar.jar`). The ADF Library JAR for the new user interface artifacts (such as pages) should be placed in the *`ExploadedWarDirectory`*`/WEB-INF/lib` directory.

For instructions on creating ADF Library, see the section on "Packaging a Reusable ADF Component into an ADF Library" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

12

Customizing and Extending SOA Components

This chapter describes how to customize (edit) service-oriented architecture (SOA) components during runtime in a deployed composite with a runtime tool such as Oracle BPM Worklist, Oracle SOA Composer, or Oracle Enterprise Manager Fusion Applications Control or customize and extend (create) SOA components during design time in Oracle JDeveloper. It also provides recommendations for merging runtime customizations from a previously deployed revision into a new revision and instructions for synchronizing customized flexfields in the Oracle Metadata Services (MDS) Repository.

This chapter includes the following sections:

For information about troubleshooting SOA issues, see the "Troubleshooting Oracle SOA Suite" chapter of *Oracle Fusion Applications Administrator's Troubleshooting Guide*.

Notes:

- This chapter does not describe customizing and extending Oracle BPM Suite. Oracle BPM Suite is installed on top of Oracle SOA Suite, and provides the ability to run Business Process Management Notation (BPMN) processes. To accomplish this task, there are extensions to JDeveloper for working with BPMN (Oracle BPM Studio) and a web-based application for working with BPMN processes (Oracle Business Process Composer). For information about BPMN process flows, see Chapter 7, "Customizing and Extending BPMN Processes." For information about Oracle BPM project templates, see Chapter 13, "Customizing and Extending Oracle BPM Project Templates."
- Oracle SOA Suite extensions cannot be used with the Integrated WLS. If an application has references to Oracle SOA Suite shared libraries, then customizations on the application cannot be tested with the Integrated WLS.

12.1 About Customizing and Extending SOA Components

SOA provides an enterprise architecture that supports building connected enterprise applications to provide solutions to business problems. SOA enables you to develop enterprise applications as modular business web services that can be integrated and reused, resulting in a flexible, adaptable IT infrastructure. SOA separates business functions into distinct units, or services.

Oracle SOA Suite provides a complete set of service infrastructure components for designing, deploying, and managing composites. A composite is an assembly of services, service components, and references designed and deployed in a single application. Wiring between the services, service components, and references enables message communication.

Oracle SOA Suite consists of SOA components that comprise the business logic and processing rules in a SOA composite. You can include components such as the following in a SOA composite:

- Business rules:

  The following categories of rules are available:

  – Approval configuration (expirations, escalations, and notifications) and assignment rules:

    Define complex task routing slips for approval management by taking into account business documents and associated rules to identify the approval hierarchy for a work item. Additionally, approval management lets you define multistage approvals with associated list builders based on supervisor or position hierarchies. You can also define expiration, escalation, and notification configurations. For example, an expense approval task may use rules to define its approvers.

    Approval configuration and assignment rules are within the context of a human workflow.

  – Nonapproval business rules:

    Define a business decision based on rules that enables dynamic decisions to be made at runtime that automate policies, computations, and reasoning while

separating rule logic from underlying application code. For example, you can define a business rule to select a supplier with the lowest shipping price to fulfill a shipping order.

Nonapproval business rules are in the context of Oracle SOA Suite, but outside of human workflow.

– Rules in non-Oracle SOA Suite applications

Use of standalone rules in non-Oracle SOA Suite applications is supported. You can completely control how the rule dictionaries are structured and how these applications are patched. You may structure the rules as recommended for Oracle SOA Suite rules, as described in this chapter.

- Domain value maps:

  Operate on actual data values that move through the infrastructure at runtime. Domain value maps enable you to map from one vocabulary used in a given domain to another vocabulary used in a different domain. For example, one domain can represent a city with a long name (Boston), while another domain can represent a city with a short name (BO). In such cases, you can directly map the values by using domain value maps.

- Human tasks:

  Extend a workflow that describes the tasks for users or groups to perform as part of an end-to-end business process flow. For example, a vacation request workflow is assigned to a manager. The manager must act on the request task three days before the vacation starts. If the manager formally approves or rejects the request, the employee is notified with the decision. If the manager does not act on the task, the request is treated as rejected. Notification actions similar to the formal rejection are taken.

- BPEL processes:

  Integrate a series of business activities and services into an end-to-end process flow. For example, a BPEL process flow calls a credit rating service. When you run this process, you enter a social security number into a user interface. The credit rating service takes the number and returns a credit rating.

- Oracle Mediators:

  Define services that perform message and event routing, filtering, and transformations. For example, Oracle Mediator can accept data contained in a text file from an application or service, transform it into a format appropriate for updating a database that serves as a customer repository, and then route and deliver the data to that database.

For more information about these components, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Oracle SOA Suite supports the following types of customizations and extensions of these components:

- Customizing several components during runtime
- Customizing and extending several components during design time

The tool to use depends on the component you are customizing or extending and whether you are performing these tasks during runtime or design time. Table 12–1 provides details.

Note: If you are customizing approval configuration and assignment rules or nonapproval business rules for a *deployed* project (either for Oracle SOA Suite or Oracle BPM Suite), always use Oracle BPM Worklist or Oracle SOA Composer. If you are customizing approval configuration and assignment rules or nonapproval business rules as part of a *new* Oracle BPM Suite project being extended in Oracle Business Process Composer, then use Oracle Business Process Composer. For information about using Oracle Business Process Composer, see Chapter 13, "Customizing and Extending Oracle BPM Project Templates."

Table 12–1 Customization and Extension Tools for Oracle SOA Suite

To Perform These Tasks...	Use This Tool...	Use This Tool At...	Tool User
Customize business rules:			
■ Approval configuration and assignment rules	■ Oracle BPM Worklist (recommended) or Oracle SOA Composer	Runtime in a deployed composite	Technical analyst
■ Nonapproval business rules	■ Oracle SOA Composer **Note**: If you use Oracle SOA Composer to customize approval configuration and assignment rules during runtime, changes in subsequent revision patches are *not* applied successfully.	Runtime in a deployed composite	Business analyst
Customize domain value maps	Oracle SOA Composer	Runtime in a deployed composite	Business analyst
Customize composite endpoint properties such as the following: ■ Attached Oracle Web Services Manager (OWSM) security policies ■ Service and reference binding component properties	Oracle Enterprise Manager Fusion Applications Control	Runtime in a deployed composite	System administrator
■ Customize or extend business rules ■ Customize or extend BPEL processes ■ Customize or extend human tasks ■ Customize or extend Oracle Mediators ■ Customize composite components such as binding components and wires ■ Customize or extend transformations ■ Extend WSDLs or XSDs ■ Extend business rules ■ Extend JCA adapters	JDeveloper (when logged in with the **Customization Developer** role)	Design time (when complete, you must deploy the composite)	System integrator

Notes:

- You *cannot* customize human tasks, BPEL processes, and Oracle Mediators during runtime in a deployed composite.
- When using Oracle SOA Composer, you can save your customizations in a sandbox environment without applying them to a running instance and later return to the sandbox to make additional customizations. These customizations are only applied to the running instance when you click **Commit**.

12.1.1 Before You Begin Customizing SOA Composites

Before you customize SOA components, become familiar with the Oracle Fusion application architecture that enables customizations, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." Also understand the typical workflows for working with runtime customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

In addition, you will need to perform the following tasks before you can begin customizing your application:

- Install JDeveloper and set up your development environment. Before you can implement customizations using JDeveloper, you must create a workspace that imports the necessary parts of the application you want to customize. For more information, see the "Setting Up Your Development Environment" chapter of *Oracle Fusion Applications Developer's Guide*.
- Create a customization application workspace. For more information, see Chapter 10, "Using JDeveloper for Customizations."
- Start JDeveloper in the appropriate role.

  For more information, see Chapter 10, "Using JDeveloper for Customizations."

12.2 Customizing SOA Composites

As described in Table 12–1, you can customize SOA components during runtime in a deployed composite with a runtime tool. This section provides an overview of these tasks and provides references to additional documentation for more specific instructions.

Note: You cannot customize SOA components in CRM Application Composer. However, you can extend a business event in CRM Application Composer and use the **Event notification** action to notify a SOA composite by email of the occurrence of the event. For information about extending events in CRM Application Composer, see Section 4.2, "Editing Objects."

Task: Start the Runtime Customization Tool

Use a web browser to start the tools shown in Table 12–2 for customizing approval configuration and assignment rules, nonapproval business rules, domain value maps, and composite endpoint properties at runtime:

Table 12–2 Starting the Customization Tool

For Customizing...	Start...	By Entering...
Business rules		
▪ Approval configuration and assignment rules	▪ Oracle BPM Worklist (recommended) ▪ Oracle SOA Composer **Note**: If you use Oracle SOA Composer to customize approval configuration and assignment rules during runtime, changes in subsequent revision patches are *not* applied successfully.	`http://host:port/integration/worklistapp` `http://host:port/soa/composer`
▪ Nonapproval business rules	Oracle SOA Composer	`http://host:port/soa/composer`
Domain value maps	Oracle SOA Composer	`http://host:port/soa/composer`
Composite endpoint properties such as OWSM security policies and binding component properties	Oracle Enterprise Manager Fusion Applications Control	`http://host:port/em`

Task: Select the Data to Customize

After accessing the runtime customization tool to use, select the data to customize.

- Oracle SOA Composer:

  1. From the **Open** list in Oracle SOA Composer, select the data to customize, as shown in Figure 12–1.

Figure 12–1 Open Menu of Oracle SOA Composer

Table 12–3 describes the options available for selection.

Table 12–3 Selecting the Data to Customize

For Customizing...	Select...
Nonapproval business rules	**Open Rules**
Domain value maps	**Open DVM**

Table 12–3 (Cont.) Selecting the Data to Customize

For Customizing...	Select...
Approval configuration and assignment rules	**Open Task** **Note**: If you use Oracle SOA Composer to customize approval configuration and assignment rules during runtime, changes in subsequent revision patches are *not* applied successfully.

- Oracle BPM Worklist:
  1. In the **Administration** section, click the **Task Configuration** tab.
  2. Select a specific approval configuration and assignment rule task to customize.

     The **Event Driven** and **Data Driven** tabs are now accessible.
  3. Select a task to view or customize from the list of task types.
- Oracle Enterprise Manager Fusion Applications Control:
  1. In the navigation pane in Oracle Enterprise Manager Fusion Applications Control, expand the **SOA** folder.
  2. Expand **soa-infra**.
  3. Expand the partition in which the SOA composites are deployed (for example, **default**).
  4. Select the SOA composite to customize.

Task: Customize Business Rules

Two categories of rules are available:

- Approval configuration and assignment rules:

  You can customize approval configuration and assignment rules included in a deployed composite in Oracle BPM Worklist (recommended), as shown in Figure 12–2, or in Oracle SOA Composer, as shown in Figure 12–3.

  For more information, see the following:

  - The "Using Approval Management Features of the Oracle BPM Worklist and Workspace" section of *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* (for Oracle BPM Worklist)
  - The "Working with Tasks at Run Time" section of *Oracle Fusion Middleware User's Guide for Oracle Business Rules* (for Oracle SOA Composer)

Figure 12–2 Approval Configuration and Assignment Rule Customizations in Oracle BPM Worklist

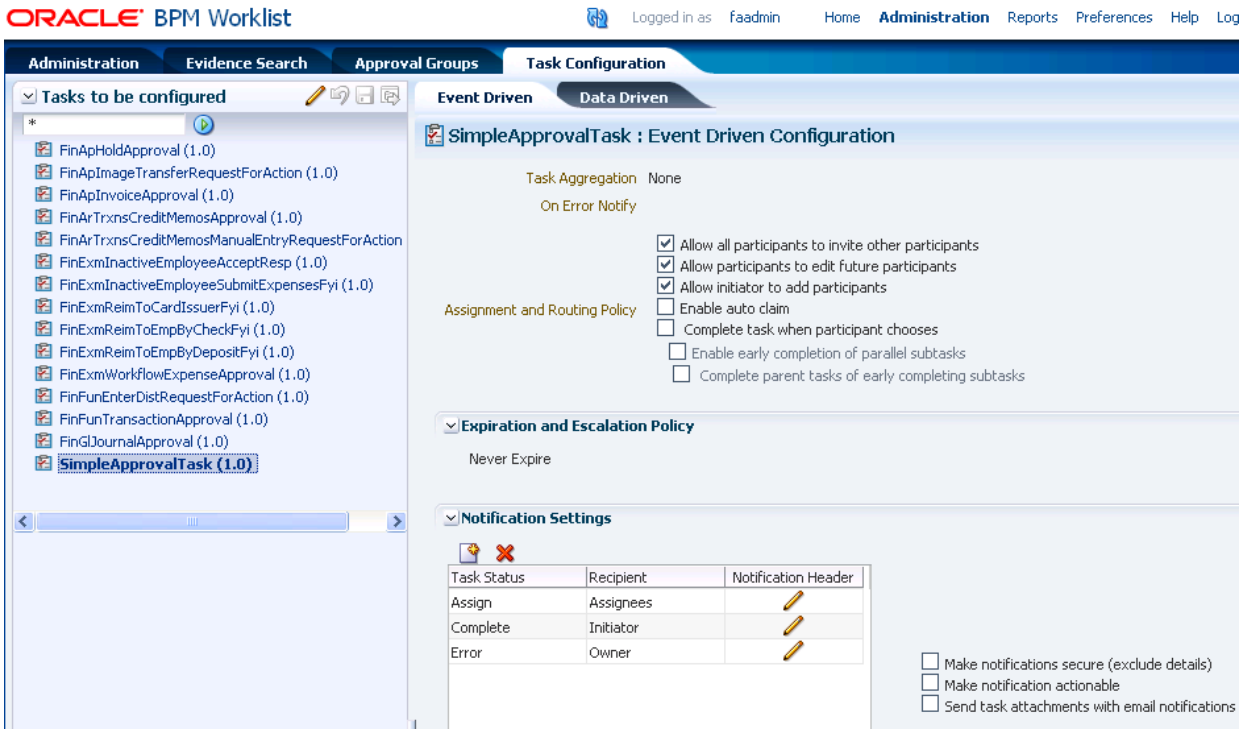
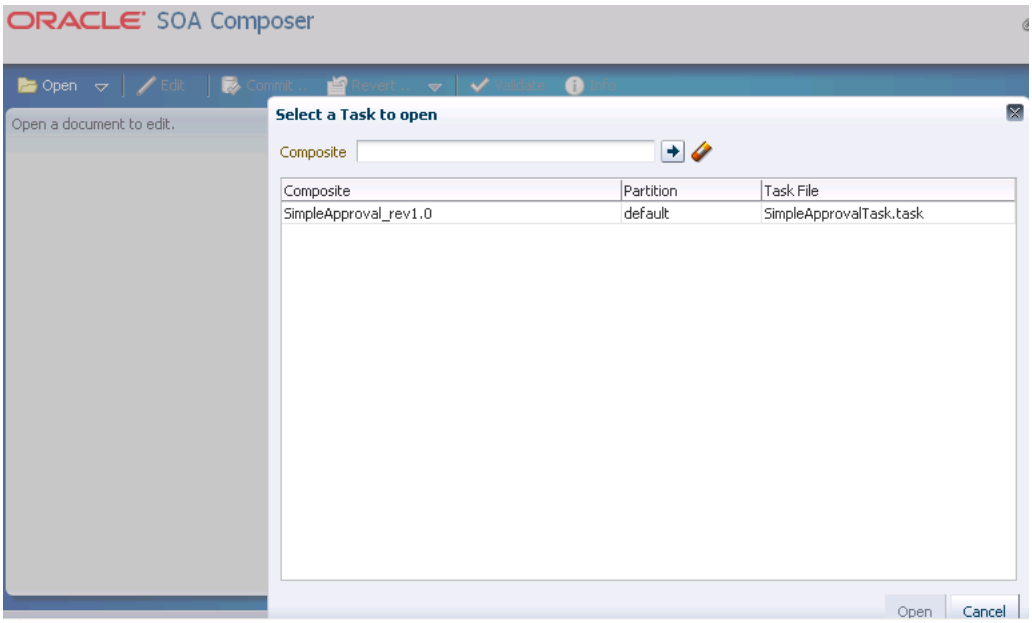


Figure 12–3 Approval Configuration and Assignment Rule Customizations in Oracle SOA Composer



How to customize the text in notifications in Oracle BPM Worklist is decided by what you want to customize in the task detail page (the page rendered when you click the task in Oracle BPM Worklist):

– Some strings are part of Oracle SOA Suite, other strings are part of the Oracle Fusion Applications-owned Oracle Application Development Framework

(ADF) resource bundle, and other strings are part of the Oracle Fusion Applications-owned SOA resource bundle.

- Task title, task outcome, approval reason, stage name, and participant type strings are stored in the Oracle Fusion Applications-owned SOA resource bundles. You *cannot* customize these because there is no support for that functionality in Oracle SOA Suite.
- Business object-specific text and sections are implemented in ADF and resource bundles are owned by Oracle Fusion Applications. These strings can only be customized in JDeveloper.
- The Oracle SOA Suite-owned strings correspond to those in the **Comments**, **Attachment**, and **History** sections in Oracle BPM Worklist. The actions along the top of the page (excluding the custom actions defined in the `.task` file) are also part of Oracle SOA Suite. These strings in the Oracle SOA Suite-owned resource bundles can be customized by following the instructions in the "Resource Bundles in Workflow Services" section of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

How text appears in email notifications for human tasks is also decided by what you want to customize:

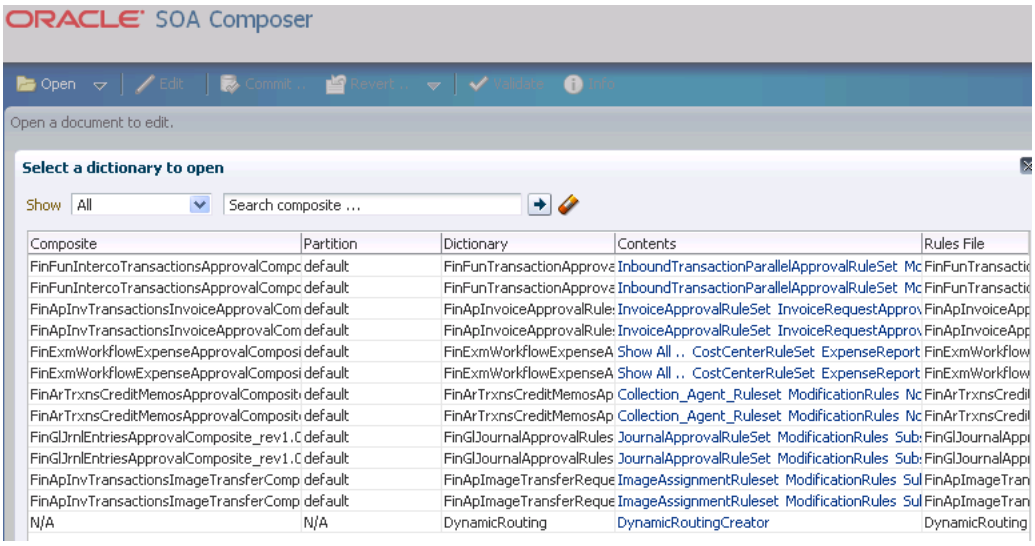
- The subject (derived from the task title) and custom outcomes are defined in the Oracle Fusion Applications-owned SOA resource bundle. You cannot customize these because there is no support for that functionality in Oracle SOA Suite.
- You can customize the notification message (the first line of instructions in the email) during runtime in Oracle BPM Worklist.
- The remaining email content is the same as customizing the text in notifications in Oracle BPM Worklist.

- Nonapproval business rules:

  You can view, customize, and commit changes to a rule dictionary included in a deployed composite in Oracle SOA Composer, as shown in Figure 12–4. Supported customizations consist of the following:

  - Customizing dictionary bucketsets
  - Customizing rules in a ruleset
  - Customizing advanced rule settings
  - Customizing conditions and actions
  - Customizing advanced mode rules
  - Customizing decision tables
  - Validating rule dictionaries

Figure 12–4 Nonapproval Business Rule Customizations in Oracle SOA Composer

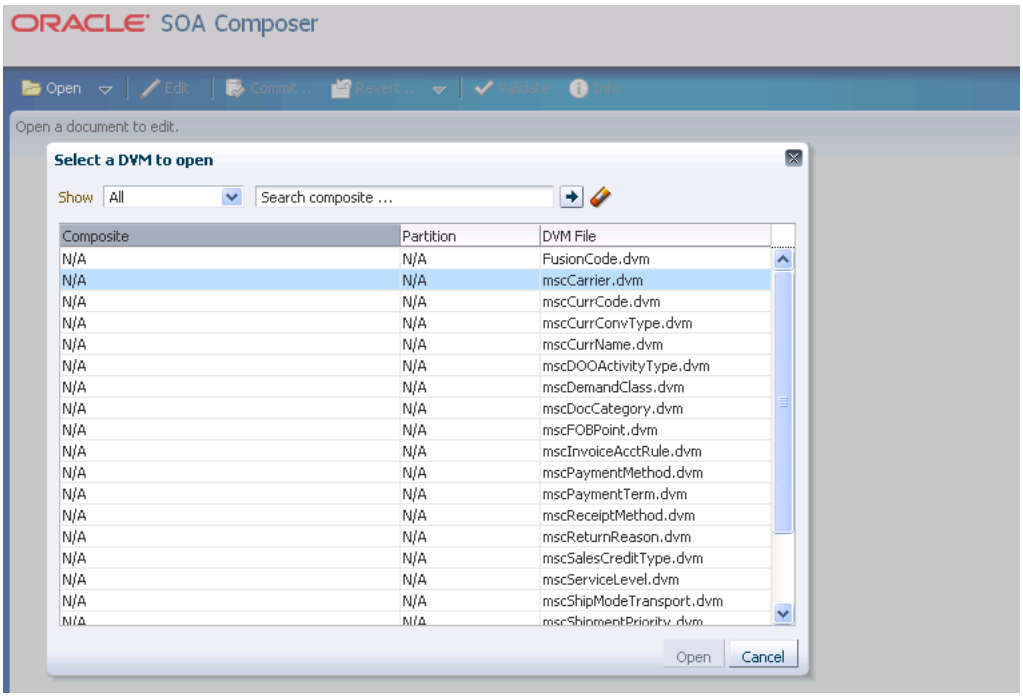


For more information about customizing business rules in Oracle SOA Composer, see the "Using Oracle SOA Composer with Oracle Business Rules" chapter of *Oracle Fusion Middleware User's Guide for Oracle Business Rules*.

Task: Customize Domain Value Maps

You can customize domain value map rows included in a deployed composite in Oracle SOA Composer, as shown in Figure 12–5. For more information, see the "Using Oracle SOA Composer with Domain Value Maps" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Figure 12–5 Domain Value Map Customizations in Oracle SOA Composer



Task: Customize Composite Endpoint Properties

You can customize endpoint address properties for an external reference such as OWSM security policies and binding components included in a deployed composite in Oracle Enterprise Manager Fusion Applications Control.

Figure 12–6 provides details about customizing OWSM security policies. For more information, see the "Managing SOA Composite Application Policies" section of *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Figure 12–6 Security Policy Customizations in Oracle Enterprise Manager Fusion Applications Control

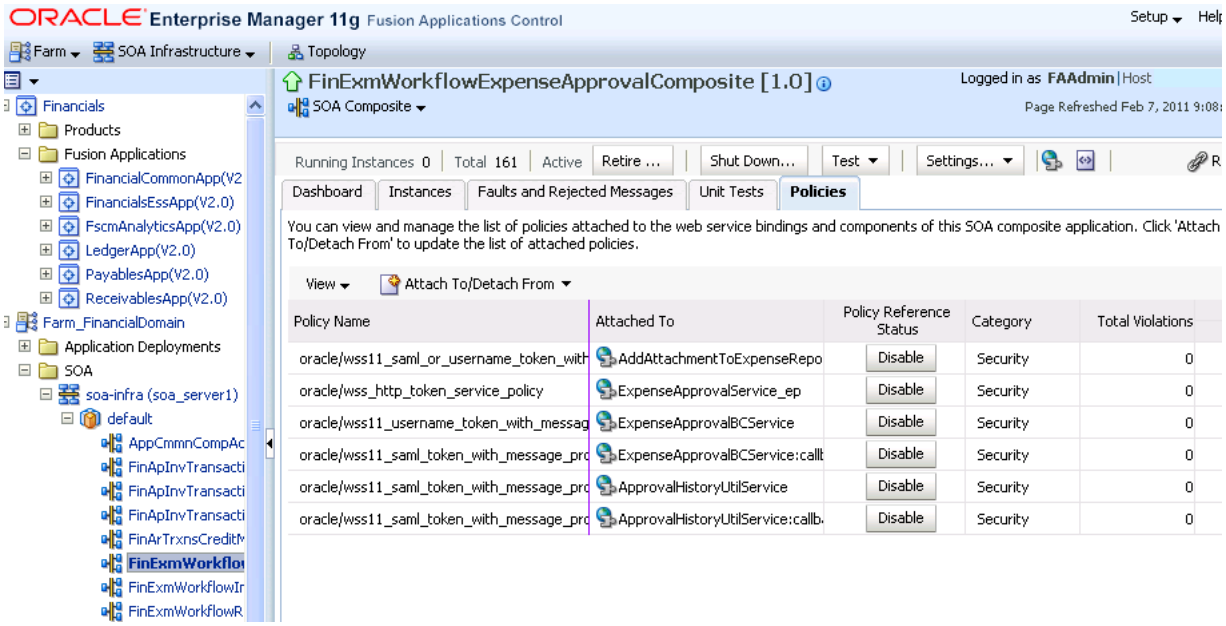
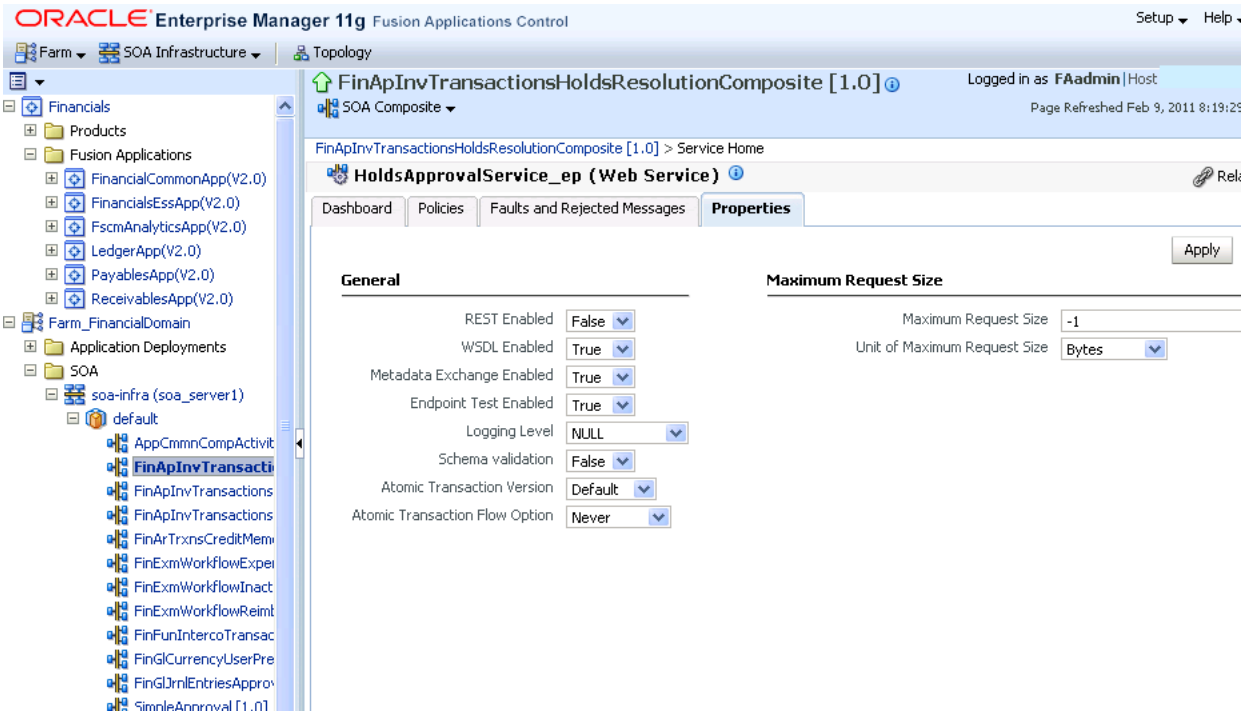


Figure 12–7 provides details about customizing binding component properties for services and references. For more information, see the "Configuring Service and Reference Binding Component Properties" chapter of *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Figure 12–7 Binding Component Property Customizations in Oracle Enterprise Manager Fusion Applications Control



Task: Synchronizing Customized Flexfields in the SOA MDS Repository

SOA composites in Oracle Fusion Applications reference copies of the original XSD schema files included in the SOA MDS Repository. When you customize and deploy Oracle Fusion Applications flexfields (or upgrade the base table, after which the flexfields are automatically re-applied), which result in a new XSD being generated in the Oracle Fusion Applications MDS Repository, the updated XSD files must be synchronized in the SOA MDS Repository for use in the fact models in business rules.

To perform this synchronization, a special SOA composite named **UpdateSOAMDS** is included with Oracle Fusion Applications. By default, the **UpdateSOAMDS** composite is automatically deployed. When a synchronization is required, you manually invoke an instance of this composite to synchronize the updated XSD files in the SOA MDS Repository. You can view the results of this synchronization in the audit trail in Oracle Enterprise Manager Fusion Applications Control.

1. Invoke the **UpdateSOAMDS** composite.
    a. Log in to Oracle Enterprise Manager Fusion Applications Control.
    b. In the navigation pane, expand **soa-infra** and the domain.
    c. Select the **UpdateSOAMDS** composite.
    d. At the top of the Dashboard page for **UpdateSOAMDS**, click **Test**.
    e. In the **Operation** list, select the operation to perform, as shown in Figure 12–8.

Figure 12–8 Operations to Perform

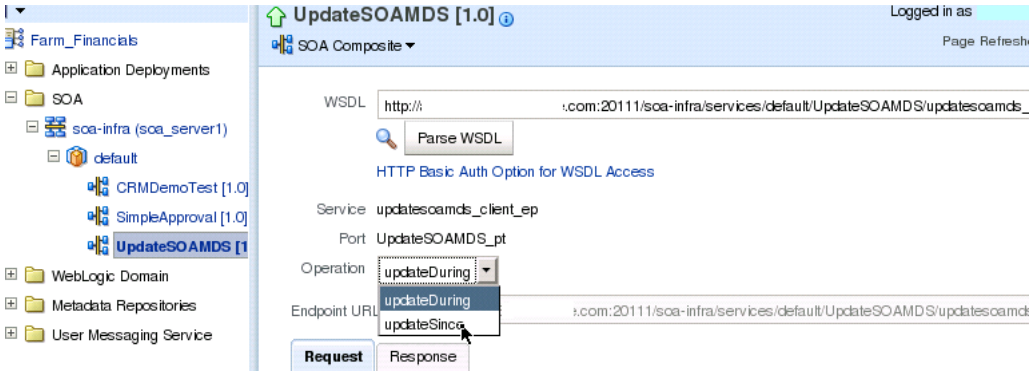


Table 12–4 describes the operations available for selection.

Table 12–4 Operations

Operation	Description
updateDuring	Select to specify how far back in time to go to get flexfield updates for synchronizing in the SOA MDS Repository.
updateSince	Select to specify the time from which you want to get flexfield updates for synchronizing in the SOA MDS Repository.

f. In the **Value** field of the **Input Arguments** section, enter a value, as shown in Figure 12–9.

Figure 12–9 Value Field

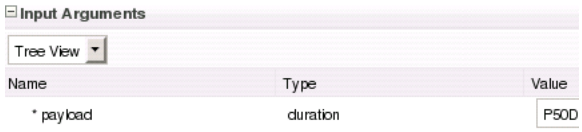


Table 12–5 provides details about the value to specify.

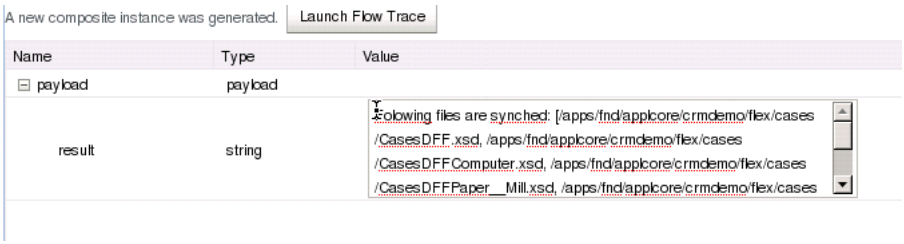
Table 12–5 Operation Values

If You Selected...	Description
updateDuring	This operation uses the `xsd:duration` type as input to obtain the data. For example: ▪ Enter `P50D` to go back fifty days to get flexfield updates that occurred. ▪ Enter `P1M2DT3H` to go back one month, two days, and three hours to get flexfield updates that occurred.
updateSince	This operation uses the `xsd:dateTime` type as input to obtain the data. For example, entering `2011-02-10T00:00:00` gets flexfield updates that have occurred since February 10, 2011.

g. Click **Test Web Service**.

The list of XSD schema files synchronized in the SOA MDS Repository is returned as output in the **Response** tab, as shown in Figure 12–10.

Figure 12–10 Test Output



All rule dictionaries in the SOA MDS Repository that use the impacted XSD schemas are altered. The data model of the rule dictionaries is modified and the fact types are re-imported. After re-importing the XSD schemas, the rule dictionary is saved in the SOA MDS Repository.

The JAXB 2.0 classes for the fact type model of the rule dictionaries that have been altered are regenerated and compiled into the appropriate composite `SCA-INF/gen-classes` directories.

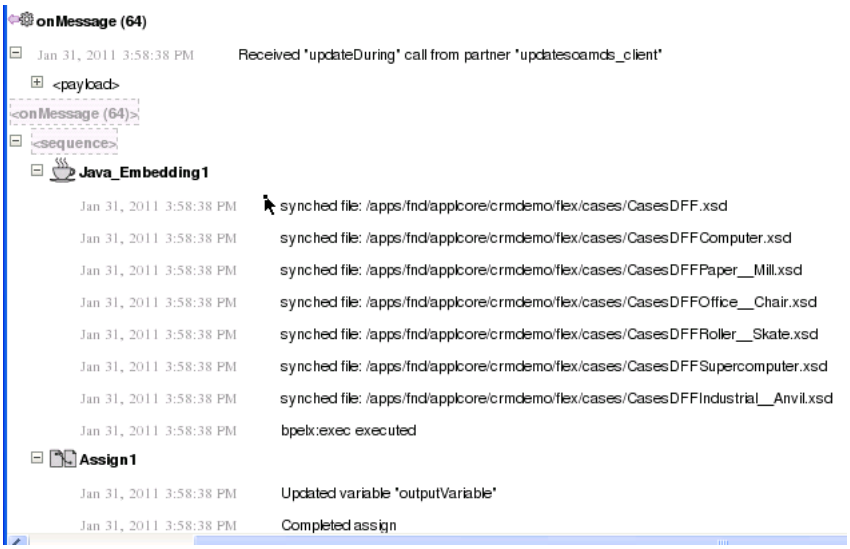
Other SOA instances in the cluster are notified of the flexfield customizations.

The composite class loader for the SOA composites where the rule dictionary was altered is invalidated and a new class loader is extended with the next request for the composite.

The SOA instances not involved in updating the rule dictionary in the SOA MDS Repository regenerate the JAXB 2.0 classes for the composites that comprise a rule dictionary where the fact type model was altered.

2. View the results in the audit trail.
   a. In the **Recent Instances** section of the Dashboard page, click the instance ID.
   b. In the **Trace** section of the Flow Trace page, click the **UpdateSOAMDS** BPEL service component.
   c. View the list of XSD schema files synchronized in the SOA MDS Repository in the audit trail, as shown in Figure 12–11.

Figure 12–11 Audit Trail Results



Note: In previous releases, Oracle BPM Worklist included a feature known as flex fields. Starting with Release 11*g* R1 (11.1.1.4), flex fields are now known as mapped attributes. Do *not* confuse Oracle BPM Worklist flex fields with Oracle Fusion Applications flexfields; they are completely different features.

12.3 Merging Runtime Customizations from a Previously Deployed Revision into a New Revision

After using a composite customized at runtime for a period of time, a new patch revision of the composite may become available. Repeating the process of customizing the new revision of the composite at runtime can be cumbersome and time-consuming. To avoid these challenges, use the OPatch tool. The OPatch tool is an Oracle-supplied, Java-based utility that enables you to merge customizations made during runtime in a previously deployed composite into a new patch revision of the composite. OPatch preserves your runtime customizations and prevents you from having to re-enter the customizations again for the next patch revision.

OPatch merges a new patch revision into a composite that was previously customized during both design time in JDeveloper and runtime in Oracle SOA Composer, Oracle BPM Worklist, or Oracle Enterprise Manager Fusion Applications Control. For specific procedures on patching SOA composites with OPatch, see the "Patching Service-Oriented Architecture (SOA) Composites" section of *Oracle Fusion Applications Patching Guide*.

Task: Merge Runtime Customizations from a Previously Deployed Revision into a New Revision

Before using the OPatch tool to merge runtime customizations from a previously deployed revision into a new revision, review the recommendations in Table 12–6 to ensure that you merge customizations successfully.

Table 12–6 Recommendations on Merging Patch Revision Customizations and Extensions

Component	Recommendation
Deletion of base components	Delete only components you added as part of the customization, and not components that are part of the base revision. This is because the deletion of base components does not survive the move to the new revision, but the deletion of the wiring does. If you delete an existing base component, it comes back again when you get the new revision, which still has the component. However, the wire deletion survives the upgrade because the `composite.xml` file is customizable.

Table 12–6 (Cont.) Recommendations on Merging Patch Revision Customizations and Extensions

Component	Recommendation
Business rules	If business rules are customized at runtime, and those runtime customizations must be preserved in subsequent revisions of the base version of the composite, it is recommended that the rules dictionaries be split into two dictionaries and linked using the dictionary linking functionality. The base rule, linked dictionary contains the data model of the dictionary and the custom rules dictionary contains all the rules customized at runtime. The OPatch process preserves the customized dictionary when it merges the customized application with subsequent versions of the application. Business rules are used in different scenarios and the following information identifies how to handle each situation. ■ Approval configuration and assignment rules These rules are used within human tasks to identify approvers and the routing of human tasks. Approval rules are always generated as base and custom dictionaries at design time. No further configuration is necessary at design time. Runtime customizations: If you must customize approval configuration and assignment rules during runtime, use only Oracle BPM Worklist to perform this task. Using Oracle BPM Worklist enables: -) Approval assignment and configuration rules to automatically be stored in a custom rules dictionary (`Rule.rules`). The custom rules dictionary is initially shipped with only sample, preseeded rules. The custom rules dictionary is separate from the base rule, linked dictionary (`RuleBase.rules`). The base rule, linked dictionary contains Oracle Fusion Applications fact definitions. Revision patches are applied to the base rule, linked dictionary. -) Changes in subsequent revision patches to be applied successfully to the base rule, linked dictionary. If you use Oracle SOA Composer to customize approval configuration and assignment rules during runtime, changes in subsequent revision patches are *not* applied successfully. Design time customizations: You cannot customize existing rules that are part of the base version of the composite at design time in JDeveloper. However, you can extend new rules that you later customize. ■ Nonapproval business rules These rules are used directly in processes like BPEL and BPMN outside of the context of a human task. These dictionaries are not generated as linked dictionaries in JDeveloper and must be manually split as linked dictionaries. Runtime customizations: If the dictionaries are split as linked dictionaries, ensure that only the linked dictionaries are customized from Oracle SOA Composer. Identification of the base rule and linked rule dictionary is up to you to develop. Design time customizations: You cannot customize existing rules that are part of the base version of the composite at design time in JDeveloper. However, you can extend new rules that you later customize.

Table 12–6 (Cont.) Recommendations on Merging Patch Revision Customizations and Extensions

Component	Recommendation
Default URLs for service binding components	Use default URLs for service binding components. If the revision is used in the URL for service binding components, when the SOA composite is patched using OPatch, the revision of the composite is customized. In this case, the reference to URLs for service binding components fails to work. In this scenario, you must manually update all the URL references for service binding components.
Oracle BPEL Process Manager scope activity	If a base composite team removes the scope activity in the next revision of the composite, when a vertical composite team or customer runs the OPatch utility to apply the new revision of the composite to their customized version, all customizations they performed on that scope activity in the first revision are lost.
Renaming of a composite whose SOA archive (SAR) file is imported in JDeveloper	When importing a SAR file for customization in JDeveloper, the composite must *not* be renamed. In addition, if you rename a composite, OPatch cannot detect runtime customizations made in Oracle SOA Composer, Oracle BPM Worklist, and Oracle Enterprise Manager Fusion Applications Control. You must manually re-apply those customizations.
Base revision of a composite with JDeveloper customizations	Assume you customize the base revision of a composite with the **Customization Developer** role in JDeveloper, and then deploy the composite. When the base revision is updated and a newer revision is made available, the customer uses OPatch to apply the patch revision. OPatch may then fail because there are JDeveloper customizations in the deployed composite. To resolve this issue, perform the following steps: 1. Open the customized composite with the **Default Role** in JDeveloper. 2. Import the patched base version 2 SAR file into this composite project extended in Section 10.3, "Customizing SOA Composites with JDeveloper." 3. Restart JDeveloper with the **Customization Developer** role. 4. Open the above customized composite. Error messages are shown in case of conflicts. 5. Resolve the conflicts in the composite. 6. Deploy the composite to the SAR file. The new SAR file should be replaced by the patched base version 2 SAR file. 7. Proceed with the OPatch process. **Note:** Ensure that the backup of the SAR files is taken properly.

12.4 Extending or Customizing Custom SOA Composites

You can customize or extend some SOA components during design time in JDeveloper when logged in with the **Customization Developer** role. Components that are extended in JDeveloper can be further customized in JDeveloper when again logged in with the **Customization Developer** role. Customization changes are maintained in separate `.xml` files from the base component files. Note that you cannot make customizations in Source view in JDeveloper; only customizations made in Design view are supported.

Notes:

- New artifacts extended in the composite survive patching.
- Ensure that you provide unique names for any new components and artifacts that you extend. For example, prefix each component and artifact name with a unique identifier.

Table 12–7 describes which existing base composite artifacts in a composite can be customized and which new artifacts can be extended when logged in to JDeveloper with the **Customization Developer** role.

Table 12–7 Customizable and Extendable Artifacts in Customization Developer Role

Artifacts	Existing Artifact in Base Composite is Customizable with Customization Developer Role?	Artifact is Extendable with Customization Developer Role?
Composite components	Yes	Yes
BPEL process	Yes	Yes
Oracle Mediator	Yes	Yes
Human task	No	Yes
Business rule	No	Yes
XSLT Map	No	Yes
Cross references (XREFs)	No	No
Domain value maps	No	No
XSD	No	Yes
WSDL	No	Yes
Business events	No	Yes
JCA Adapters	No	Yes

Table 12–8 provides more specific details about which artifacts can be extended when logged in to JDeveloper with the **Customization Developer** role.

Table 12–8 Artifact Extensibility in JDeveloper with Customization Developer Role

Artifact	Extendable?	Description
Composite	No	Only one composite per Oracle SOA Suite project is permitted.
BPEL process	Yes	Can drag a BPEL process from the Component Palette into the SOA Composite Editor or Oracle BPEL Designer.
Oracle Mediator	Yes	Can drag an Oracle Mediator from the Component Palette into the SOA Composite Editor or Oracle BPEL Designer.
Human task	Yes	Can drag a human task from the Component Palette into the SOA Composite Editor or Oracle BPEL Designer.

Table 12–8 (Cont.) Artifact Extensibility in JDeveloper with Customization Developer

Artifact	Extendable?	Description
Business rule	Yes	Can drag a business rule from the Component Palette into the SOA Composite Editor or Oracle BPEL Designer.
XSLT map	Yes	Can extend a transformation in a transform activity in Oracle BPEL Designer or the Mediator Editor.
Domain value maps	No	The New Gallery dialog is disabled with the **Customization Developer** role.
Cross references (XREFs)	No	The New Gallery dialog is disabled with the **Customization Developer** role.
XSD	Yes	Right-click an Oracle SOA Suite project and select **SOA**, or as the result of extending other SOA artifacts.
WSDL	Yes	Right-click an Oracle SOA Suite project and select **SOA**, or as the result of extending other SOA artifacts.
Business events	Yes	Subscribe to or publish events for a BPEL process or Oracle Mediator component in the SOA Composite Editor, Oracle BPEL Designer, or Mediator Editor.
JCA adapters	Yes	Drag adapters from the Component Palette into the SOA Composite Editor or Oracle BPEL Designer.

Task: Customize a Base Composite in JDeveloper

This section provides an overview of the steps for customizing a base composite of Oracle Fusion Applications in JDeveloper. These steps assume you know how to set up the customization layer through the `adf-config.xml` editor and know the customization classes defined by Oracle Fusion Applications. For more information, see the "Customizing SOA Composite Applications" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

> **Note:** See Section 10.3.2, "Setting Up the JDeveloper Workspace and Composite Project for MDS Repository Customization" for instructions on setting up the JWS workspace and composite project when customizing Oracle Fusion Applications composites.

1. Install Oracle Fusion Applications with a SOA composite that you want to customize in JDeveloper.
2. In Oracle Enterprise Manager Fusion Applications Control, go to the home page of the SOA composite to export.
3. From the **SOA Composite** menu at the top of the page, select **Export**.
4. Obtain the base SAR file for initially customizing from other locations, including:
   - Checking out the base composite project from the source control system where the base composite project was checked in by the base development team. This way, no SAR file deployment, export command, or import command is involved.
   - Importing the base composite SAR file that was deployed from the base composite project.
   - Importing the base composite SAR file that was exported (without runtime changes) from the Export Composite page of the Oracle Enterprise Manager

Fusion Applications Control installation from which the SOA server is managed.

5. Extend layer values for customization to the `CustomizationLayerValues.xml` file (can perform this task in JDeveloper or from the directory structure).
6. Start JDeveloper in the **Default Role**.
7. Extend a new composite application.
8. From the **File** main menu, select **Import** > **SOA Archive Into SOA Project** to import the exported SAR into the new composite in JDeveloper.
9. In the Import Composite Archive wizard, select the **Import For Customization** checkbox.
10. From the **Tools** main menu, select **Preferences** > **Roles** > **Customization Developer**.
11. Restart JDeveloper, and customize the layers of the composite.
12. Right-click the project and select **Deploy** to extend a customized SAR of the SOA composite in Oracle Fusion Applications.

Note: After performing the initial customizations described in these procedures, you can no longer export the composite from the runtime. This is because the composite is a merged composite, and no longer the original base composite.

For more information on exporting SAR files, see the "Exporting a Deployed SOA Composite Application" section of *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Task: Extend or Customize Custom Business Rules

You can extend business rules in a composite during design time in JDeveloper when logged in with the **Customization Developer** role. After extending these business rules, you can further customize them in JDeveloper when again logged in with the **Customization Developer** role. You cannot customize existing business rules that are part of the base version of the composite.

For information on customizing business rules during runtime, see Section 12.2, "Customizing SOA Composites."

Task: Extend or Customize Custom BPEL Processes

You can extend or customize BPEL processes in a composite during design time in JDeveloper when logged in with the **Customization Developer** role. For example, you can perform the following tasks:

- Extend or delete a new scope or other activity
- Customize an activity
- Extend a partner link
- Extend a transformation

For more information about extending or customizing BPEL processes, see the "Using the BPEL Process Service Component" part of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Extend or Customize Custom Human Tasks

You can extend human tasks in a composite during design time in JDeveloper when logged in with the **Customization Developer** role. After extending these human tasks, you can further customize them in JDeveloper when again logged in with the **Customization Developer** role. You cannot customize existing human tasks that are part of the base version of the composite.

For more information on extending human tasks, see the "Using the Human Workflow Service Component" part of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Extend or Customize Custom Oracle Mediators

You can extend or customize Oracle Mediators in a composite during design time in JDeveloper when logged in with the **Customization Developer** role. For example, you can perform the following tasks:

- Extend a routing rule
- Customize an XPath condition
- Make any other changes, except those that impact files such as XSLs (for transformations), WSDLs, EDLs (for business events), or XSDs. Note that *new* artifacts can be extended or customized.

For more information, see the "Using the Oracle Mediator Service Component" part of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Customize Composite Components

You can customize composite endpoint properties in a composite during design time in JDeveloper when logged in with the **Customization Developer** role. For example, you can perform the following tasks:

- Extend and delete a reference binding component
- Extend and delete a service binding component (entry point)
- Extend, customize, and delete a wire between components

For more information, see the "Developing SOA Composite Applications with Oracle SOA Suite" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Extend or Customize Transformations (in a Transform Activity)

You cannot customize existing transformations that are part of the base composite in JDeveloper. However, you can extend a new transform activity in a BPEL process or in the Transformation Map dialog of an Oracle Mediator during design time in JDeveloper when logged in with the **Customization Developer** role. After extending the transformation, you can further customize it in JDeveloper when again logged in with the **Customization Developer** role. For example, you can perform the following tasks:

- Specify the mapper file (`.xsl`) to which the transform activity points from the **Mapper File** field of a transform activity in a BPEL process or the Transformation Map dialog of an Oracle Mediator. However, you cannot extend or customize mappings. The mappings are defined in the XSL file (not in the transform activity), which is not customizable.
- Copy an out-of-the-box XSL file into a custom XSL artifact, add the custom logic to the custom XSL, and customize the transform activity to reference the custom XSL. Additionally, you must copy the contents of the XSL file in the base composite into the custom XSL file.

Task: Extend XSD or WSDL Files

You can extend an XSD schema or WSDL document in JDeveloper when logged in with the **Customization Developer** role.

1. Right-click the Oracle SOA Suite project in the Application Navigator.
2. Select **SOA**.
3. Select the artifact to extend:
   - **Create XML Schema**

     Invokes the Create XML Schema dialog for extending a new XML schema file in the project. When complete, the new schema file automatically opens.
   - **Create WSDL Document**

     Invokes the Create WSDL dialog to extend a new WSDL file in the project.

Task: Extend Business Events

You cannot directly extend business events in JDeveloper when logged in with the **Customization Developer** role. The **New Gallery** dialog that is displayed when you select **New** from the **File** main menu is disabled with the **Customization Developer** role. However, you can create business events as part of other Oracle SOA Suite customizations such as when allowing an Oracle Mediator to subscribe to an event.

For more information, see the "Using Business Events and the Event Delivery Network" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Extend JCA Adapters

You can extend JCA adapters in JDeveloper when logged in with the **Customization Developer** role.

For more information, see *Oracle Fusion Middleware User's Guide for Technology Adapters*.

12.5 Deploying SOA Composite Customizations and Extensions

You must redeploy a customized or extended composite after making changes in JDeveloper. The development and deployment phase is as follows:

- During base composite development, you create a customizable SOA project from the **Default** role in JDeveloper, set up customization layers, and deploy the composite to a base SAR file.
- During customization, you import (for customization) the base composite SAR file to extend a new SOA project, change from the **Default** role to the **Customization Developer** role, perform customizations, and deploy the composite to create a customized SAR file.

For more information, see the "Customizing SOA Composite Applications" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

12.6 Extending a New Oracle SOA Suite Service

You can extend new SOA composite services to integrate with Oracle Fusion Applications. This section provides an overview of tasks for extending and consuming new services and provides references to documentation that more specifically describes these tasks.

Task: Setting Up a Development Environment

You must set up and configure a development environment in JDeveloper to create new Oracle SOA Suite services. For more information, see the "Getting Started Building Your Oracle Fusion Applications" part of *Oracle Fusion Applications Developer's Guide*.

Task: Using JDeveloper to Create Applications, Projects, and Services

Whenever you create new projects, you must first create an application using templates provided by JDeveloper. For more information, see the "Setting Up Your JDeveloper Workspace and Projects" chapter of *Oracle Fusion Applications Developer's Guide*.

You can select an Oracle SOA Suite project template when creating a JDeveloper application. For more information about creating Oracle SOA Suite projects, see the "Developing SOA Composite Applications with Oracle SOA Suite" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

You can extend an Oracle ADF Business Component service to be consumed by the SOA composite. The Oracle ADF Business Component service is used for connecting Oracle ADF applications using service data object (SDO) data formats with the SOA composite. For more information, see the "Getting Started with Binding Components" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Understanding Common Service Use Cases and Design Patterns

There are fundamental patterns for Oracle Fusion Applications developers to follow when building applications involving Oracle ADF and Oracle SOA Suite. These patterns fall into three main categories:

- Using business events to initiate business processes
- Orchestrating over business logic implemented with Oracle ADF, Java, PL/SQL, and SOA composite applications
- Modeling human task flows in Oracle ADF applications

For more information about these and other design categories, see the "Common Service Use Cases and Design Patterns" part of *Oracle Fusion Applications Developer's Guide*.

Task: Using Oracle SOA Suite with the Oracle Metadata Services Repository

The Oracle MDS Repository contains metadata for certain types of deployed applications, such as SOA composites. You can store Oracle Fusion Applications artifacts and custom artifacts in the Oracle MDS Repository. You connect to the repository to consume these artifacts.

For more information about the Oracle MDS Repository, see the "Managing the Metadata Repository" chapter of *Oracle Fusion Middleware Administrator's Guide*.

For more information about creating a connection from Oracle SOA Suite to the Oracle MDS Repository, using the Oracle SOA Suite MDS Repository to store custom artifacts, and connecting to and consuming artifacts from the Oracle SOA Suite MDS Repository, see the "Creating a SOA-MDS Connection" section of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Task: Discovering Oracle Fusion Applications Services

Oracle Fusion Applications includes web services that are available for public consumption. These web services are defined in Oracle Enterprise Repository and available for discovery. When extending Oracle Fusion Applications and building

SOA composites to invoke services built by Oracle Fusion Applications, you can use Oracle Enterprise Repository to perform the following tasks:

- Use Oracle Enterprise Repository to discover the service.
- Follow the link provided by Oracle Enterprise Repository to access the WSDL file.
- When building the client, have JDeveloper download the WSDL file locally so that the client is not accessing the runtime WSDL file.

For more information about Oracle Enterprise Repository, see *Oracle Fusion Middleware User Guide for Oracle Enterprise Repository*.

Task: Securing Oracle Fusion Applications and Services

You must secure Oracle Fusion Applications and services to be consumed by SOA composites.

For more information about Oracle Fusion Applications security, see *Oracle Fusion Applications Security Guide*.

For more information about ADF Application Artifacts security, see Chapter 15, "Customizing Security for ADF Application Artifacts."

For more information about web services security, see the "Securing Web Services Use Cases" chapter of *Oracle Fusion Applications Developer's Guide*.

Task: Deploying SOA Composites and Services

You must deploy SOA composites and the services to be consumed.

For more information about deploying SOA composites, see the "Deploying SOA Composite Applications" chapter of *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

For more information about deploying external references such as web services, see the "Deploying Web Services Applications" chapter of *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

Task: Understanding Fusion Applications Deployment Topology

An enterprise deployment is an Oracle guidelines blueprint based on proven Oracle high-availability and security technologies and recommendations for Oracle Fusion Applications. For more information about deployment in an enterprise environment, see *Oracle Fusion Applications Customer Relationship Management Enterprise Deployment Guide*.

13

Customizing and Extending Oracle BPM Project Templates

This chapter describes how to use Oracle JDeveloper to customize and extend Oracle BPM project templates. BPM projects contain the Business Process Modeling Notation (BPMN) processes used by Oracle Fusion applications. Several Oracle Fusion applications use BPMN processes to define process flows within the application.

This chapter includes the following sections:

13.1 About Customizing Project Templates

BPM project templates are templates used to create new BPM projects. Project templates are created by developers and contain all of the elements necessary to create a new BPM project that can be deployed to runtime. This includes all of the necessary BPMN processes and business catalog components.

Oracle Fusion applications provide default project templates containing the required BPMN processes and business catalog components. Refer to the product-specific documentation in Oracle Fusion Applications Help for a list of the default BPM project templates provided by Oracle Fusion applications.

Developers can customize and extend these project templates. Project templates are customized or extended by developers using Oracle BPM Studio is an extension to JDeveloper that provides additional editors for creating and customizing BPMN processes and related components.

For more information on Oracle BPM Studio see the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

In the context of Oracle Fusion applications, developers can customize project templates when it is necessary to customize or extend business catalog components that are part of the default project templates. Refer to the product-specific documentation in Oracle Fusion Applications Help for a list of the default BPM project templates provided by Oracle Fusion applications.

After customizing or extending a project template, it can be published to the Oracle BPM repository. Project templates are shared between Oracle BPM Studio and Oracle Business Process Composer using the Oracle BPM repository. Additionally, BPM projects can be shared between Business Process Composer and JDeveloper users via the BPM repository.

Note: When customizing a project template, you must first make a copy of the existing template using JDeveloper. This enables you to avoid overwriting project templates previously saved to the BPM repository.

After a template is published to the repository, it is available to Business Process Composer users. Business Process Composer users can create and edit BPM projects created using these templates and can collaborate on these projects with process developers using JDeveloper. They can also create SAR files and deployment plans or deploy BPM projects directly to the BPM runtime environment without having to reedit and deploy a project using JDeveloper.

See the "Workflow: Creating Project Templates" section in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for information on the typical workflow for sharing project templates between Oracle BPM Studio and Business Process Composer.

13.1.1 About the Business Catalog

The business catalog is a set of reusable components that contain all of the necessary technical implementation to create a BPMN process flow that can be deployed as part of a running Oracle Fusion application.

The business catalog contains the following components:

- Business rules

  Define a business decision based on rules that enables dynamic decisions to be made at runtime that automate policies, computations, and reasoning while separating rule logic from underlying application code.

- Human tasks

  Create a workflow that describes the tasks for users or groups to perform as part of an end-to-end business process flow.

- Services

  Define how a BPMN process connects to other business processes and systems, including databases and web services.

Some elements of the business catalog can be customized using Oracle Business Process Composer. See Section 7.1.2.2, "What You Can Customize Using Business Process Composer" for information on those elements.

13.1.2 Before You Begin Using JDeveloper to Customize Project Templates

Before you customize the artifacts within a project template, including business catalog components, business processes, SOA components using JDeveloper, you should be familiar with the Oracle Fusion applications architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications"

You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle"

In addition, you will need to do the following before you can use JDeveloper to customize BPM project templates:

- Download and install JDeveloper, and set up your development environment, as documented in the "Setting Up Your Development Environment" chapter of the *Oracle Fusion Applications Developer's Guide*.
- Launch JDeveloper in the default role.

13.2 Customizing or Extending a Project Template

This section outlines the general tasks you must perform to customize or extend an Oracle BPM project template.

Task: Open a Project Template

You can open a project template with Oracle BPM Studio.

- For information on opening a project template, see the "Working with Project Templates" section in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management.*

The specific project or project template you need to open depends on which Oracle Fusion application you are customizing. Refer to the product-specific documentation in Oracle Fusion Applications Help for a list of the default BPM project templates provided by Oracle Fusion Applications.

Task: Create or Customize BPMN Processes

BPMN processes are accessible using the BPM Project Navigator. For information on using the BPM Project Navigator, see the "Oracle BPM Project Navigator" section in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

For information on opening a BPMN process, see the "How to Open a Business Process" section in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

See the "Working with Flow Objects in Your Process" section of the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on working with flow objects in your process.

See the "Modeling Business Processes with Oracle BPM" section of the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for general information on BPMN flow objects.

Task: Create or Modify Business Catalog Components

Using Oracle BPM Studio, you can create or modify business catalog components within a project template.

- Services
- Human tasks
- Business Rules

Task: Customize SOA Components

BPM projects are based on technology provided by the Oracle SOA Suite. This includes reusable components and services that are included as part of a project template.

In addition to customizing business catalog components, you can customize applications by customizing SOA components, including the following:

- Domain value maps
- BPEL processes
- Oracle Mediators

See Section 12.4, "Extending or Customizing Custom SOA Composites" for more information.

13.3 Publishing Project Templates to the BPM Repository

In Oracle BPM, publishing a project template refers to the process of saving it in the Oracle BPM repository. You can publish project templates to the repository to make them available to Business Process Composer users.

The repository can also be used to share BPM projects between Business Process Composer and JDeveloper users as part of the process development life-cycle.

Publishing a project template to the BPM repository makes them available to Business Process Composer user enabling collaboration between application developers and business users.

Task: Configure an Oracle BPM MDS Connection

Before publishing a project template to the Oracle BPM MDS repository, you must configure an MDS connection.

See the "How to Configure a Connection to the Oracle BPM Metadata Service Repository" section in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on creating a connection to the repository.

Task: Publish a Project Template

See the "How to Publish a Project or Project Template to Oracle BPM MDS" section of the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for information on publishing a project template.

After publishing a project template, it is available to Business Process Composer users who can use it to create new BPMN process flows. See Chapter 7, "Customizing and Extending BPMN Processes" for more information.

14

Customizing and Extending Oracle Enterprise Scheduler Jobs

This chapter describes how to use Oracle JDeveloper or Oracle Enterprise Manager Fusion Applications Control to create and extend scheduled jobs using Oracle Enterprise Scheduler.

This chapter includes the following sections:

14.1 About Customizing and Extending Oracle Enterprise Scheduler Jobs

Enterprise applications require the ability to respond to many real-time transactions requested by end users or web services. However, they also require the ability to offload larger transactions to run at a future time, or automate the running of application maintenance work based on a defined schedule.

Oracle Enterprise Scheduler provides the ability to run different job types, including: Java, PL/SQL and spawned processes, distributed across nodes in a server cluster. Oracle Enterprise Scheduler runs these jobs securely, and provides monitoring and management through Fusion Applications Control.

Oracle Enterprise Scheduler provides scheduling services for the following purposes:

- Distributing job request processing across a cluster of servers,
- Running Java, PL/SQL and binary jobs,
- Scheduling job requests based on recurrence,
- Managing job requests with Fusion Applications Control.

Oracle Enterprise Scheduler provides the critical requirements in a service-oriented environment to automate processes that must recur on a scheduled basis and to defer heavy processing to specific time windows. Oracle Enterprise Scheduler lets you:

- Support sophisticated scheduling and workload management,
- Automate the running of administrative jobs,

- Schedule the creation and distribution of reports,
- Schedule a future time for a step in a business flow for business process management.

14.1.1 Main Steps for Extending Oracle Enterprise Scheduler Jobs

Extending Oracle Enterprise Scheduler jobs involves the following main steps:

1. Develop the code that implements the job logic.
2. Create a metadata file for the job definition.
3. Grant permissions to the job, such that only those with the proper permissions can request job submission.
4. Enable job request submission, using an existing hosting application, a pre-configured user interface or a new customized application.

14.1.2 Main Steps for Customizing Oracle Enterprise Scheduler Jobs

Customizing Oracle Enterprise Scheduler jobs involves editing job properties using Oracle Enterprise Manager Fusion Applications Control. The job properties that you can modify are described in Table 14–8.

14.1.3 Before You Begin Extending and Customizing Oracle Enterprise Scheduler Jobs

Before you extend and customize Oracle Enterprise Scheduler jobs, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflow for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can begin extending Oracle Enterprise Scheduler jobs:

- For developers:

  Set up JDeveloper. For more information, see Section 1.3.2, "Installing Customization Tools."

- For administrators:
  - Install Oracle Fusion Applications, making sure to provision Oracle Enterprise Scheduler services. For more information, see the *Oracle Fusion Applications Installation Guide*.
  - Start Fusion Applications Control. For more information about starting up and using Fusion Applications Control, see the chapter "Getting Started with Managing Oracle Fusion Applications" in *Oracle Fusion Applications Administrator's Guide*.

14.2 Extending Custom Oracle Enterprise Scheduler Jobs Using Existing Oracle Fusion Applications

There are two main use cases for creating Oracle Enterprise Scheduler jobs.

Oracle Enterprise Scheduler Administrator

Administrators can create a new job definition using Oracle Enterprise Manager Fusion Applications Control console, using an existing hosting application. Scheduled

jobs typically required by administrators include database maintenance tasks using PL/SQL or running spawned jobs or scripts such as SQL*Plus scripts to load data into the database. Once you have defined the job, use Oracle Enterprise Manager Fusion Applications Control to submit the job request.

Developer or System Integrator

When using an existing hosting application, use Fusion Applications Control to create Oracle Business Intelligence Publisher, PL/SQL and spawned jobs. Use JDeveloper to create Java jobs and develop a new hosting application that executes a set of jobs. A Java job might invoke an Oracle ADF Business Components service or a SOA composite, for example.

In cases where there is no need to repackage the hosting application, PL/SQL, binary, Oracle BI Publisher and Java jobs can be added to any hosting application. Optionally, you can execute Java jobs from a custom hosting application.

System integrators may want to use Fusion Applications Control to develop a job, while developers may prefer JDeveloper. Jobs are typically submitted using the scheduled request submission UI. Alternatively, it is possible to develop an Oracle ADF application with screens for submitting Oracle Enterprise Scheduler jobs.

Task: Implement the Logic for the Oracle Enterprise Scheduler Job

An Oracle Enterprise Scheduler job is a request to execute a specific task written in code or a script, such as Java, PL/SQL, spawned jobs, and so on.

An example of logic to be implemented by a scheduled job is writing particular data to a database under certain conditions, for example, daily shift schedules for a given sub-set of employees.

Task: Create a Job Definition Metadata File

An Oracle Enterprise Scheduler job definition specifies the type of job to be run (such as Java, PL/SQL type jobs, and so on), the hosting application that will run the job, and any additional required or optional parameters and properties for the job.

It is possible to create a job definition in Oracle Enterprise Manager Fusion Applications Control or JDeveloper.

The minimum required properties and parameters for each job type are as follows:

- Oracle BI Publisher jobs: Specify the `reportid` parameter. Specify Oracle BI Publisher parameters as job parameters with required validation. These can be entered by end users during request submission using the request submission user interface.

  If the report is a bursting report, identify it as such by selecting the bursting check box.

- PL/SQL jobs: In the job definition, specify the PL/SQL procedure that includes the job logic implementation.

  All input arguments to the PL/SQL procedure can be specified as parameters with required validation. These can be entered by end users during request submission using the request submission user interface.

- All other job types: Specify the name of the implementation logic and parameters in the job definition.

For more information about creating a job definition in Oracle Enterprise Manager Fusion Applications Control, see the chapter "Managing Oracle Enterprise Scheduler Service and Jobs" in *Oracle Fusion Applications Administrator's Guide*.

For more information about creating a job definition in JDeveloper, see the chapter see "Working with Extensions to Oracle Enterprise Scheduler" in *Oracle Fusion Applications Developer's Guide*.

Task: Grant Relevant Permissions

Grant the appropriate permissions for the application using the Oracle Authorization Policy Manager.

An example of the use of relevant permissions is to grant execution permissions to a role so that users belonging to that role can submit the job.

For more information about granting relevant permissions to a deployed application, see *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Enable Job Request Submission

You can enable job request submissions through an Oracle ADF user interface using JDeveloper or Fusion Applications Control.

When using JDeveloper to enable job request submissions through an Oracle ADF user interface, you must define a view object to capture properties filled in by end users.

If a job is defined with properties that must be filled in by end users, the user interface allows end users to fill in these properties prior to submitting the job request. For example, if the job requires a start and end time, end users can fill in the desired start and end times in the space provided by the user interface.

The properties that are filled in by end users are associated with a view object, which in turn is associated with the job definition itself. When the job runs, Oracle Enterprise Scheduler accesses the view object to retrieve the values of the properties.

You could, alternatively, submit job requests using Fusion Applications Control. Using Fusion Applications Control to enable job request submissions through an Oracle ADF user interface does not require you to create a view object for capturing end user data.

> **Note:** Suppose a parameter view object is VLinked to another view object (VO1). If you customize the view object using JDeveloper, then the Oracle Enterprise Scheduler job submission UI list of values reflects this customization, if the customization is in the MDS runtime database.

For more information about submitting job requests using Fusion Applications Control, see the chapter "Managing Oracle Enterprise Scheduler Service and Jobs" in *Oracle Fusion Applications Administrator's Guide*.

For more information about defining a view object for use with a job submission interface, see the chapter "Working with Extensions to Oracle Enterprise Scheduler" in *Oracle Fusion Applications Developer's Guide*.

14.2.1 Extending a Custom PL/SQL Oracle Enterprise Scheduler Job

Extending a custom PL/SQL Oracle Enterprise Scheduler job involves creating a PL/SQL package and defining job metadata.

Task: Implement the Logic for the PL/SQL Job

Implementing a PL/SQL scheduled job involves creating a PL/SQL package and defining the job metadata using the PL/SQL job type.

To implement the logic for a PL/SQL job:

1. Create a PL/SQL package, including the required `errbuf` and `retcode` arguments. A sample PL/SQL package is shown in Example 14–1.

Example 14–1 Sample PL/SQL package

```
CREATE OR REPLACE PACKAGE XxSamplePkg AUTHID CURRENT_USER AS

Procedure SampleJob (
     errbuf out NOCOPY varchar2,
     retcode out NOCOPY varchar2,
     name in varchar2 );

END XxSamplePkg;
/

CREATE OR REPLACE PACKAGE BODY XxSamplePkg AS

Procedure SampleJob (
 errbuf out NOCOPY varchar2,
 retcode out NOCOPY varchar2,
 name in varchar2 )
 IS

begin
      -- Write log file content using the FND_FILE API.
      FND_FILE.PUT_LINE(FND_FILE.LOG, 'Running Stored procedure
SampleJob.........');
      FND_FILE.PUT_LINE(FND_FILE.LOG, 'FND USERNAME : ' || FND_GLOBAL.USER_NAME);

      -- Write log file content using the FND_FILE API.
      FND_FILE.PUT_LINE(FND_FILE.OUTPUT,' Name : ' || name );
      FND_FILE.PUT_LINE(FND_FILE.OUTPUT, 'Job Request id : ' || FND_JOB.REQUEST_ID
);

      errbuf := fnd_message.get_string('FND', 'COMPLETED NORMAL');
      retcode := 0;

  end SampleJob;

END XxSamplePkg;
/
```

2. Deploy the package to Oracle Database.
3. Grant the required permissions, and perform any other necessary tasks in the database.

   ```
   grant execute on xxSampleJob to FUSION_APPS_EXECUTE;
   ```

   For more information about granting permissions for the execution of a PL/SQL job, see the section "Performing Oracle Database Tasks for PL/SQL Stored Procedures" in the chapter "Creating and Using PL/SQL Jobs" in *Oracle Fusion Applications Developer's Guide*.
4. Test the package.

Task: Create a Job Definition Metadata File for the PL/SQL Job

Use the Setup and Maintenance work area to define a job definition metadata file for the PL/SQL job. The job definition metadata file may also include user properties for the PL/SQL job as well as UI parameters to be displayed at runtime.

For more information about creating an Oracle Enterprise Scheduler metadata file, see the section "Creating a Job Definition" in the chapter "Managing Oracle Enterprise Scheduler Service and Jobs" in *Oracle Fusion Applications Administrator's Guide*.

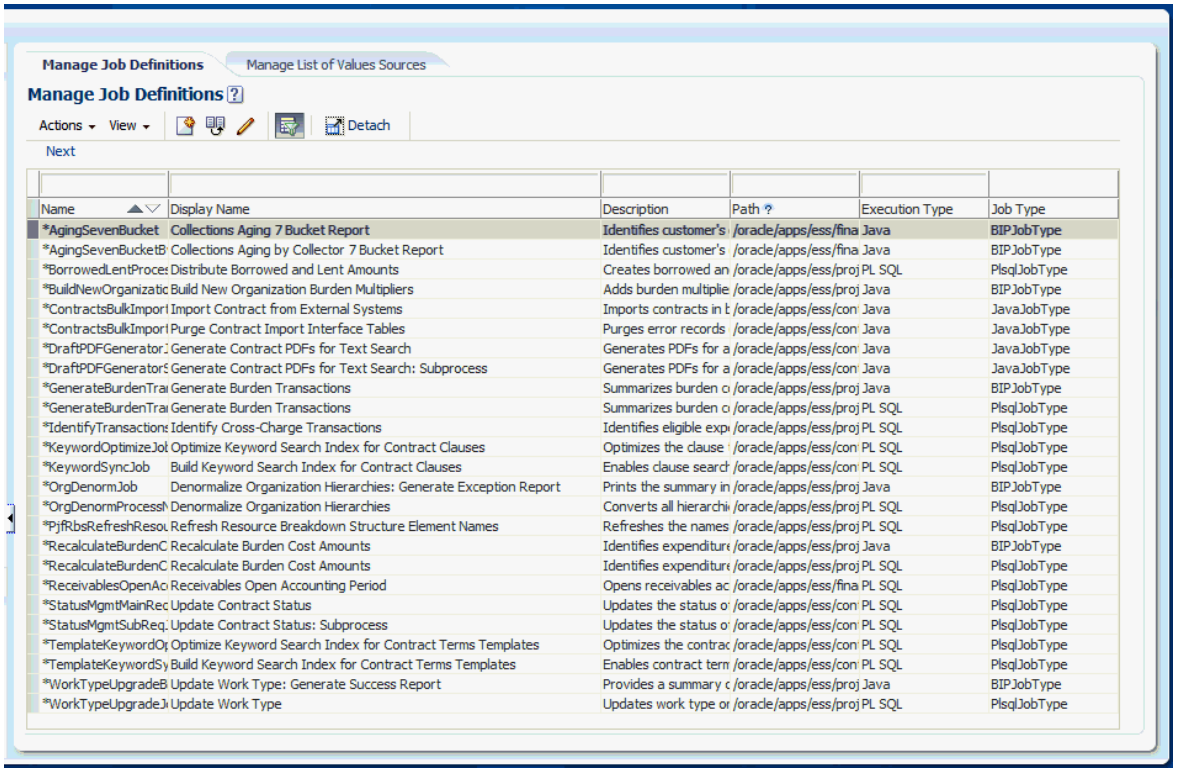
To create a job definition metadata file for a PL/SQL job:

1. From the Administration menu in the global area of Oracle Fusion Applications, select the **Setup and Maintenance** work area and click the **All Tasks** tab. Search for all tasks.
2. From the list of tasks that displays, select the relevant UI application you will use to host the job definitions and parameter view objects. This Oracle Fusion application is the portlet producer application for the job.

   Click the **Go to Task** button.

   The Manage Job Definitions tab displays, as shown in Figure 14–1.

Figure 14–1 The Manage Job Definitions tab



3. In the Manage Job Definitions tab, click the **New** button.
4. In the Create Job Definition tab, click **Show More** to display all parameters and enter the values for the job shown in Table 14–1.

Table 14–1 PL/SQL Job Definition Values

Field	Description
Display Name	Enter a display name for the job.
Name	Enter a name for the job definition.
Path	Specify the trailing package name for the job definition metadata. The default namespace or path for custom job definitions begins with `oracle/apps/ess/custom`. For example, when entering `test` in the **Path** text field, the job definition is stored in the `globalEss` MDS namespace as `oracle/apps/ess/custom/test`.
Job Application Name	From the dropdown list, select the name of the hosting application running the Oracle Enterprise Scheduler job.
Job Type	Select the job type from the dropdown list, namely the **PlsqlJobType**.
Procedure Name	Enter the name of the stored procedure to run as part of the PL/SQL job execution.
Standard request submission flag	Check this box to indicate that the job request is to be submitted in the standard manner.

5. At the bottom of the pane, click the **User Properties** tab. Define the following user properties by clicking the **New** button, as shown in Table 14–2.

Table 14–2 PL/SQL User Properties

Name	Data Type	Default Value	Read Only
`EXT_ PortletContainerWebModule`	String	For the default value, enter the name of the web module which will be used as a portlet when submitting the job request.	N/A
`numberOfArgs`	String	Set the number of job submission arguments, including `errbuf` and `retcode`.	N/A

Note: Typically, the web context is registered as the web module name. Verify with your applications administrator the value of the registered web module name in the Topology Manager work area. Registering the correct web module name enables the correct remote rendering of the Oracle Fusion application job request parameters from the Oracle Enterprise Scheduler central UI.

6. Click the ***<Job Definition Name>*: Parameters** tab and specify UI parameters as required. The UI parameter fields are described in Table 14–3.

Table 14–3 PL/SQL Job UI Parameters

Field	Description
Prompt	Enter the text to be displayed at the prompt that displays during run time.
Data Type	From the dropdown list, select the relevant data type.
Page Element	From the dropdown list, select the UI page element you want to use to display the parameter, for example, a text box.

7. Click **Save and Close** to create and save the new Oracle Enterprise Scheduler PL/SQL job definition.

14.2.2 Extending a Custom Oracle BI Publisher Oracle Enterprise Scheduler Job

Implementing a Oracle BI Publisher scheduled job involves creating a Oracle BI Publisher report on the Oracle BI Publisher Server and defining the Oracle Enterprise Scheduler job metadata.

Task: Implement the Logic for the Oracle BI Publisher Job

For information about implementing an Oracle BI Publisher job, see the chapter "Using BI Publisher with Oracle JDeveloper" in the *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher (Oracle Fusion Applications Edition)*.

Task: Create a Job Definition Metadata File for the Oracle BI Publisher Job

Using the Setup and Maintenance work area, create an Oracle BI Publisher type job definition.

To create a job definition metadata file for an Oracle BI Publisher job:

1. Follow the instructions in "Task: Create a Job Definition Metadata File for the PL/SQL Job".
2. From the Job Type dropdown list, select **BIPJobType**.
3. In the User Properties tab, define only the **EXT_PortletContainerWebModule** property.

14.2.3 Extending a Custom Java Oracle Enterprise Scheduler Job

Implementing a Java scheduled job involves implementing the Java business logic and defining the relevant Oracle Enterprise Scheduler job metadata. Use JDeveloper to implement a Java job and deploy the job as a shared library. Modify the deployment descriptor of the deployed user interface or hosting application EAR file so that it points to the shared library. Redeploy the file.

Deploying the job as a shared library allows you to add additional jobs in the future without having to redeploy the hosting application. For more information about deploying Oracle ADF applications, see the chapter "Deploying Fusion Web Applications" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Implement the Logic for the Java Job

In order to develop an application that runs a Java class under Oracle Enterprise Scheduler, you must define the Java class that implements the Oracle Enterprise Scheduler executable interface. The executable interface defines the contract that enables using Oracle Enterprise Scheduler to invoke a Java class.

To create a Java class for an existing Oracle Enterprise Scheduler Oracle Fusion application, take the following steps:

- Create an application in JDeveloper.
- Create a project in JDeveloper.
- Develop the application code that uses the Oracle Enterprise Scheduler Java APIs.

To implement the logic for an Oracle Enterprise Scheduler Java job:

1. In JDeveloper, create an application and project. Make sure to include EJB and Java technologies in the project.
2. Add the Oracle Enterprise Scheduler extensions to the project.

a. In the Application Navigator, right-click the project you just created. Select **Project Properties** and then select Libraries and Classpath.

b. In the Libraries and Classpath pane, click **Add Library**.

c. In the Add Library window, in the Libraries field, select **Enterprise Scheduler Extensions** and click **OK**.

3. Create a Java class using the Oracle Enterprise Scheduler package.

a. In the project overview tab, click the Java Files link.

b. In the Java Files pane, click the **New** button. From the Create New in Project menu, select ***Project Name*** and then select Java Class.

The Create Java Class window displays.

c. In the Create Java Class window, enter a name for the Java class and the package name in the fields provided. For example, if working with the Financials Oracle Fusion application, the package name would be **oracle.apps.financials.ess.program**. Accept the remaining default values.

4. In the Java class, develop the code that will do the work of the Java job. Example 14–2 shows sample code which illustrates the use of an Oracle Enterprise Scheduler job request file handle and writes a job request parameter submitted to the request log and output files.

Example 14–2 Sample Java code

```
package oracle.apps.financials.ess.program;

   import java.io.IOException;
   import oracle.as.scheduler.Cancellable;
   import oracle.as.scheduler.Executable;

   import oracle.as.scheduler.ExecutionCancelledException;
   import oracle.as.scheduler.ExecutionErrorException;
   import oracle.as.scheduler.ExecutionPausedException;
   import oracle.as.scheduler.ExecutionWarningException;
   import oracle.as.scheduler.RequestExecutionContext;

   import oracle.as.scheduler.RequestParameters;
   import oracle.as.scheduler.SystemProperty;

   import oracle.as.scheduler.cp.exec.ProcessHelper;
   import oracle.as.scheduler.cp.file.LogFile;
   import oracle.as.scheduler.cp.file.OutputFile;

   public class XxSampleJob implements Executable, Cancellable {

      private OutputFile requestOutput;
      private LogFile requestLog;

      private boolean m_isCancelled = false;

      private long request_id = 0L;
      private String requestParameter1 = null;

      public XxSampleJob() {
          super();
      }
```

```
public void execute(RequestExecutionContext ctx,
                    RequestParameters params) throws ExecutionErrorException,
                                                     ExecutionWarningException,
                                                     ExecutionCancelledException,
                                                     ExecutionPausedException {

    request_id = ctx.getRequestId();

    System.out.println("XxSampleJob Running, Request ID: " +
                       ctx.getRequestId());

    try {

        String userFileDir =
                    (String)params.getValue(SystemProperty.USER_FILE_DIR);

        String sysPropUserName =
                    (String)params.getValue(SystemProperty.USER_NAME);

        // Read the job request parameter.
        requestParameter1 = (String) params.getValue("submit.argument1");

        requestOutput = ProcessHelper.getOutputFile();
        requestOutput.writeln("Sample ESS Java job execution OUTPUT");
        requestOutput.writeln("USER_NAME as SystemProperty: " +
                              sysPropUserName);
        requestOutput.writeln("ESS Job requestID: " + request_id);
        requestOutput.writeln("ESS Job request parameter: "
                                                    + requestParameter1);

        requestLog = ProcessHelper.getLogFile();
        requestLog.writeln("Sample ESS Java job execution LOG");
        requestLog.writeln("ESS requestFileDirectory: " + userFileDir);
        requestLog.writeln("ESS Job requestID: " + request_id);
        requestLog.writeln("ESS Job request parameter: "
                                                    + requestParameter1);

    }  catch (Exception ex) {

        System.out.println("Exception running XxSampleJob: " +
                           ex.getMessage());
        ex.printStackTrace();

    } finally {

        try {

            // Close all open job request log and output files.
            ProcessHelper.closeAllFiles();

        } catch (IOException ioe) {

            System.out.println("Exception closing files: " +
                               ioe.getMessage());
            ioe.printStackTrace();
        }

    }

}
```

```
    @Override
    public void cancel() {
        m_isCancelled = true;
    }

}
```

Task: Deploy the Java Business Logic

In order to deploy the Java logic of an Oracle Enterprise Scheduler Java job, identify an existing Oracle Fusion application as the target hosting application.

Next, update the Java business logic for an existing Oracle Fusion application as follows:

- Package the Java application in a JAR file.
- Update JAR module in the Oracle Fusion application classpath.
- Bounce the server instance to load the Java program logic in the Oracle Fusion application class loader.

To deploy the Java business logic:

1. Create a deployment profile for the project.
   a. In JDeveloper, from the Application Navigator, select the project you created. Build the project to ensure that the Java class successfully compiles.
   b. Right-click the project, select **Project Properties** and **Deployment**.
   c. In the Deployment Profiles field, click **New** to create a deployment profile for the JAR file.

      The Create Deployment Profile window displays.
   d. In the Create Deployment Profile window, enter a name for the deployment profile and click **OK**.
   e. In the Edit JAR Deployment Profile Properties window, verify that the Java job class is included in the JAR module output and click **OK**.
2. Package the Oracle Enterprise Scheduler Java class into a JAR file and deploy it.
   a. From the Application Navigator, right-click the project you created. Select **Deploy** and then select the JAR file you just created.

      The Deployment Action window displays.
   b. In the Deployment Action window, click **Finish** to create a packaged JAR archive.

      The archive module is deployed to the default project deployment path, for example, **$JDEV_HOME**/<**PROJECT_NAME**>/deploy/<**JAR_NAME**>.jar.

      Note: All custom JAR files must begin with the prefix `Xx`, for example `XxMyJar.jar`.
3. Update the JAR module in the application classpath of the Oracle Enterprise Schedulerhosting application.

a. Locate the expanded deployment directory of the EAR file for the existing Oracle Fusion application, for example **$MW_HOME**/fusionapps/applications/fin/deploy/EarFinancialsEss.ear/APP-INF/lib.

b. Copy the deployed custom jar file to the expanded directory.

4. In the domain to which the Oracle Enterprise Scheduler hosting application is deployed, restart Oracle Enterprise Scheduler.

   The Oracle Enterprise Scheduler job executes the updated Java class once the application class loader successfully loads the updated class.

   For more information about restarting the Oracle Enterprise Scheduler, see the section "Starting and Stopping an Oracle Enterprise Scheduler Service Instance" in the chapter "Managing Oracle Enterprise Scheduler Service and Jobs" in *Oracle Fusion Applications Administrator's Guide*.

Task: Create a Job Definition Metadata File for the Java Job

Using the Setup and Maintenance work area, create a Java type job definition.

To create a job definition metadata file for a Java job:

1. Follow the instructions in "Task: Create a Job Definition Metadata File for the PL/SQL Job".
2. In the Create Job Definition window, from the Job Type dropdown list, select **JavaJobType**.
3. In the Create Job Definition window, in the Class Name field, enter the fully qualified class name of the Java business logic.
4. In the Create Job Definition window, In the User Properties tab, define only the **EXT_PortletContainerWebModule** property.
5. Click the **<*Job Definition Name*>: Parameters** tab and specify UI parameters as required. The UI parameter fields are described in Table 14–3.
6. Click **Save and Close** to create and save the new Oracle Enterprise Scheduler Java job definition.

14.2.4 Submitting Oracle Enterprise Scheduler Jobs

You can use Oracle Fusion Applications to submit Oracle Enterprise Scheduler jobs.

To submit Oracle Enterprise Scheduler jobs:

1. In the global area of Oracle Fusion Applications, access the Schedule Processes page by clicking the **Navigator** menu and then selecting **Tools** and **Schedule Processes**.
2. Click **Schedule New Process**.

   The Search and Select: Process Name window displays.
3. In the Process Name field, enter the name of the Oracle Enterprise Scheduler job you want to schedule and click **Search**.

   The job name displays in the search results table.
4. From the search results table, select the job name and click **OK**.

   The Process Details page displays.

5. In the Process Details page, in the Parameters field, specify any required parameters.
6. Click **Submit** to request that the Oracle Enterprise Scheduler instance execute the job. Click **Close** to return to the Scheduled Processes page.
7. In the Scheduled Processes page, refresh the Search Results table to monitor the status of submitted job.

14.3 Creating a Custom Oracle Enterprise Scheduler Application to Extend Oracle Enterprise Scheduler Jobs

Use ANT scripts to develop and deploy an Oracle Enterprise Scheduler hosting application and user interface. Use JDeveloper to create the relevant metadata.

14.3.1 Creating Hosting and User Interface Applications Using an ANT Script

Use the supplied ANT script to create the hosting and user interface applications for the Oracle Enterprise Scheduler jobs.

When deploying the application, be sure to identify the product family and use an existing registered Oracle WebLogic Server domain. This allows you to test your application without having to create and register a domain, or register any associated applications with the product family.

To create hosting and user interface applications using scripts:

1. Extract the Oracle Enterprise Scheduler `customer_extensibility` script from the JDeveloper installation or JDeveloper Extensions to the development work environment, for example, into a folder called `template_home`.

   The `template_home` folder contains an ANT `build.xml` driver file that processes the template Oracle Enterprise Scheduler hosting and producer web applications by replacing macros with specified inputs.
2. Change directories to the `template_home` directory to create the `user_home` directory that will contain the resulting macro-substituted files copied from `template_home`.
3. Run the script in any of the following ways:
   - Interactively, where you are prompted for the relevant inputs. Accept the default, if there is one, by pressing Enter at each prompt.

     In the `template_home` directory, type `ant` or ant `create-user-home`. A sample running script is shown in Example 14–3.

Example 14–3 Interactive script execution

```
$ ant
 Buildfile: build.xml
-init:

 create-user-home:
    [input] Enter which template should be used (source_template) (default=Fusion)
    [input]      ([Fusion], Standalone)
                   Fusion
    [input] Enter Middleware Home Directory path (fmw_home_dir) (default=) []
            /JDEVADF_INSTALLATION/
    [input] Enter hosting application name (hosting_application_name) (default=MyAppEss) [MyAppEss]
            MyAppEss
```

```
    [input] Enter hosting application JPS stripe id (hosting_application_stripe_id)
            (default=MyAppEss) [MyAppEss]
            MyAppEss
    [input] Enter UI application name (ui_application_name) (default=MyApp) [MyApp]
            MyApp
    [input] Enter UI application JPS stripe id (ui_application_stripe_id) (default=MyApp) [MyApp]
            MyAppEss
    [input] Enter the shared library name for the job business logic (jobdef_library_name)
            (default=MyJobsLibrary) [MyJobsLibrary]
            oracle.ess.shared
    [input] Enter an empty directory where the applications will be created (user_home)
            /workspace/ess_user_home
     [echo]
     [echo]
    [mkdir] Created dir: /workspace/ess_user_home
[propertyfile] Creating new property file: /workspace/ess_user_home/template.properties
     [copy] Copying 31 files to /workspace/ess_user_home
     [copy] Copied 36 empty directories to 14 empty directories under /workspace/ess_user_home
     [copy] Copying 19 files to /workspace/ess_user_home
     [move] Moving 1 file to /workspace/ess_user_home/Template_Hosting
     [move] Moving 1 file to /workspace/ess_user_home/Template_UI
     [echo]
     [echo] ========================================
     [echo]
     [echo] A new workspace has been created at: /workspace/ess_user_home
     [echo] This workspace can be opened and modified using JDeveloper
     [echo] To deploy the applications, run the following command:
     [echo]     ant -f /workspace/ess_user_home/ant/build-ess.xml deploy
     [echo] To create new jobs from predefined templates, run the following command:
     [echo]     ant -f /workspace/ess_user_home/build.xml create-new-job-def

BUILD SUCCESSFUL
```

- Using predefined property files. Any properties not defined in a file can be entered at the prompt. A sample properties file is shown in Example 14–4. To create a properties file, execute the command `:$> cat myProperties.properties`, where `myProperties.properties` is the name of the properties file.

Example 14–4 Script execution via property files

```
user_home=/home/myuser/ess_user_home/

 ui_application_name=MyApp
 ui_application_stripe_id=MyApp
 ui_application_version=V2.0

 hosting_application_name=MyAppEss
 hosting_application_stripe_id=MyAppEss
 hosting_application_version=V2.0

 jobdef_library_name=oracle.ess.sharedlibrary
 jobdef_library_spec_version=11
 jobdef_library_impl_version=11.1.1.5.0
```

Then execute the following command:

```
$> ant create-user-home -propertyfile myProperties.properties
```

- Specifying individual properties at the command line. Any properties not defined in a file can be entered at the prompt.

Example 14–5 Script execution via the command line

```
$> ant create-user-home -Dui_application_name=MyApp -Dhosting_application_
name=MyAppEss
```

To view supported options, enter `ant help-create-user-home` at the prompt.

4. On successful execution, you can modify the application template workspace from the `user_home` directory in JDeveloper.

   At the prompt, enter `ant help-deploy` to list the supported deployment options.

14.3.2 Generating an Oracle Enterprise Scheduler Synchronous Java Job Business Logic Template

In the event that you want to run a synchronous Java scheduled job, you will need to develop the business logic for the job. Use the build.xml file extracted in Section 14.3.1 to create a template for the business logic of the Java job.

To generate an Oracle Enterprise Scheduler Java job business logic template:

1. To create new jobs from pre-defined templates, run the following command:

   ```
   ant -f ${ess_user_home_dir}/build.xml create-new-job-def
   ```

2. When prompted, enter the Oracle Enterprise Scheduler job name, for example, `HelloSyncJavaJob`, and the package name, for example, `oracle.apps.financials.ess.program`.

   Note: Ensure that the full job package name is unique across product families.

   A sample command execution is shown in Example 14–6.

Example 14–6 Creating a Java job business logic template

```
Buildfile: /workspace/ess_user_home/build.xml

-init:

create-new-job-def:
    [echo] Available Job Definition Templates:
    [echo]     1) Simple Synchronous Java Job
   [input] Enter number of job definition template to create (job_template_to_create)
 1
    [echo] Calling default target on /my_ess_main/ess/util/customer_extensibility/Fusion/
           Template_JobLibrary/simple_synchronous_job/build.xml

-init:

create-job-definition:
   [input] Enter Java package name for Job Definition (jobdef_package_name)
           (default=oracle.apps.ess.custom) [oracle.apps.ess.custom]
           oracle.apps.financials.ess.program
```

```
   [input] Enter Java class name for Job Definition (jobdef_class_name)
           (default=MySynchronousJavaJob) [MySynchronousJavaJob]
           HelloSyncJavaJob
    [copy] Copying 1 file to /workspace/ess_user_home/MyAppEss/EssSharedLibrary/src
    [copy] Copying 1 file to /workspace/ess_user_home/MyAppEss/EssSharedLibrary/src/oracle/
           apps/financials/ess/program

BUILD SUCCESSFUL
```

3. In JDeveloper, open the Oracle Enterprise Scheduler hosting application project saved to the `user_home` workspace location.
4. In the Navigator, expand the EssSharedLibrary Model project to modify the template-generated Java job business logic.
5. Modify the file as required and save your changes.

14.3.3 Creating Fusion Oracle Enterprise Scheduler Job Metadata Using JDeveloper

In order to submit job requests using the Oracle Enterprise Scheduler hosting application, you need to create metadata that defines a job request, including the following:

- **Job type**: This specifies an execution type and defines a common set of parameters for a job request.
- **Job definition**: This is the basic unit of work that defines a job request in Oracle Enterprise Scheduler.

14.3.3.1 Creating an Oracle Enterprise Scheduler Job Definition in the Hosting Application

To use a Java class with Oracle Enterprise Scheduler you need to create a job definition. When creating a job definition, specify a name, select a job type, and specify system properties.

To create a job definition in the hosting application:

1. In the Application Navigator, right-click the EssSharedLibrary project and select **New** to display the New Gallery window.
2. In the New Gallery in the Categories area, expand Business Tier and select **Enterprise Scheduler Metadata**.
3. From the New Gallery Items area, select **Job Definition** and click **OK**.

   The Create Job Definition window displays.
4. In the Create Job Definition window, specify the following:
   - In the Name field, enter a name for the job definition. For example, for the scheduler hosting application, enter **SampleJob**.
   - In the Package field, enter a package name. For example, enter `oracle/apps/ess/custom/test`.
   - From the Job Type dropdown list, select **JavaJobType**.

   Click **OK**. The job definition `SampleJob.xml` is created, as well as the jobs folder in the package `oracle/apps/ess/custom/test`. The Job Definition page displays.
5. In the Job Definition page, specify the fully qualified class name of the template generated Java business logic created in Section 14.3.2.

6. Next to the Class Name field, select the **Overwrite** checkbox.
7. In the Class Name field, enter the name of the Java class you created, for example, `oracle.apps.financials.ess.program.HelloSyncJavaJob`.
8. In the System Properties section, click the **Add** button and create a system property called **EffectiveApplication**. Set the value of the property to the hosting application name, for example, `MyAppEss`.
9. In the Parameters section, define the following required properties:
   - **jobDefinitionName**: The short name of the job. For example, `SampleJob`.
   - **jobDefinitionApplication**: The short name of the hosting application running the job. For example, `MyAppEss`.
   - **jobPackageName**: The name of the package running the job. For example, `/oracle/apps/ess/custom/test`.
   - **srsFlag**:A boolean parameter (Y or N) that controls whether the job displays in the job request submission user interface. Enter **Y**.
   - **EXT_PortletContainerWebModule**: The name of the web module for the Oracle Enterprise Scheduler Central UI application to use as a portlet when submitting a job request. For example, `MyApp`, or any producer web application (if you prefer to use an existing registered web module that hosts the ADF view objects).
   - **parametersVO**: The ADF Business Components view object you define so that end users may enter additional properties at runtime through an Oracle ADF user interface. For example, `oracle.apps.financials.ess.SampleVO`. See Task: Create an ADF Business Components Oracle Enterprise Scheduler View Object for more information about creating a view object in the Oracle ADF producer application.

14.3.3.2 Creating a Schedule Request Submission User Interface Enabling End Users to Fill in Properties

If your job includes any properties to be filled in by end users at runtime, you need to create an Oracle ADF user interface and an ADF Business Components view object with validation and the parameters to be filled in. The Oracle Enterprise Scheduler schedule request submission user interface allows end users to fill in these properties prior to submitting the job request.

For more information about Oracle ADF view objects, see the chapter "Creating a Business Domain Layer Using Entity Objects" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Task: Create an ADF Model Project

Create an ADF model project to display the properties to be filled in by end users at runtime.

To create an Oracle ADF model project:

1. In JDeveloper, open the Oracle Enterprise Scheduler Oracle ADF application.
2. From the Application menu, select **New Project**.
3. In the New Gallery, under Categories expand **General** and select **Projects**.
4. In the Items area, select **ADF Model Project** and click **OK**.

5. On the Name Your Project wizard page, enter the project name, for example **EssModel**. Click **Finish** to close the wizard.
6. From the Application Navigator, right-click the **EssModel** project and select **Project Properties**, then **Libraries and Classpath**, and then **Add Library**.
7. Add the required data model project libraries as described in the chapter "Setting Up Your JDeveloper Workspace and Projects" in the *Oracle Fusion Applications Developer's Guide*.
8. Click **OK** to dismiss the Project Properties dialog.

Task: Create an ADF Business Components Oracle Enterprise Scheduler View Object

Use a parameters view object for jobs with parameters that require collecting values from end users at runtime. The properties filled in by end users are associated with an ADF Business Components view object, which is associated with the job definition itself. When the job runs, Oracle Enterprise Scheduler accesses the view object to retrieve the values of the properties.

To create an ADF Business Components Oracle Enterprise Scheduler view object:

1. In JDeveloper in the Application Navigator, right-click the project **EssModel** in which you want to create the view object and choose **New**.
2. In the New Gallery, expand Business Tier, select ADF Business Components and then View Object. Click **OK**.

   If this is the first component you're creating in the project, the Initialize Business Components Project dialog displays, allowing you to select a database connection.
3. In the Initialize Business Components Project dialog, select the database connection or choose **New** to create a connection.

   Click **OK**. This launches the Create View Object wizard.
4. In the Create View Object wizard on the Name page, enter the following.
   - **Package**: Enter package information for the view object, for example `oracle.apps.financials.ess`.
   - **Name**: Provide a name, for example, `SampleVO`.
   - **Select the data source type you want to use as the basis for this view object**: For the data source, select **Rows Populated Programmatically, not Based on a Query**.

   Note: Enter the view object package and name values specified for the job definition property `parametersVO` in Section 14.3.3.1.
5. Click **Next**. In the Attributes page, click **Finish** to create the Oracle Enterprise Scheduler parameter view object `SampleVO`.
6. Define attributes for the view objects sequentially, `ATTRIBUTE1`, `ATTRIBUTE2`, and so on, with an attribute for each required parameter.
7. Create a query for the view object.
   a. On the View Object page, from the left list panel select **Query**.
   b. In the Query panel, click the **Edit** icon.
   c. Use the following query and test for validity.

```
select null as ATTRIBUTE1 from dual
```

d. Click **OK**.

Note: A maximum of 100 attributes can be used for `parametersVO`. The attributes should be named incrementally, for example `ATTRIBUTE1`, `ATTRIBUTE2`, and so on. Attribute names are not case sensitive, such that `ATTRIBUTE1` and `Attribute2` can be used sequentially.

---

8. Ensure that the view object attributes are always able to be updated.
   a. On the View Object page, from the left list panel, select **Attributes**.
   b. Edit the `ATTRIBUTE1` table row.
   c. In the Edit Attribute: Attribute1 window, select the radio button option **Always**.
   d. In the Edit Attribute: Attribute1 window, click **Control Hints** to display the Control Hints page. In the Control Hints page, specify required prompts, validation, and formatting for each parameter.
   e. Click **OK**.
9. If not already specified, add the property `parametersVO` to your Oracle Enterprise Scheduler hosting application job definition and specify the fully qualified path of the view object as the value of `parametersVO`. For example, set `parametersVO` to `oracle.apps.financials.ess.SampleVO` in the job definition `/oracle/apps/ess/custom/test/SampleJob.xml`.

```
<parameter name="parametersVO"
data-type="string">oracle.apps.financials.ess.SampleVO</parameter>
```

14.3.4 Assembling Oracle Enterprise Scheduler Oracle Fusion Applications

Assembling the Oracle Enterprise Scheduler Oracle Fusion applications involves the following main steps:

- Assembling an Oracle Enterprise Scheduler shared library,
- Assembling the hosting application,
- Assembling the ADF producer application.

Task: Assemble an Oracle Enterprise Scheduler Shared Library

Assembling a shared library for Oracle Enterprise Scheduler involves the following main steps:

- Creating or updating a shared library JAR manifest.
- Updating the shared library JAR deployment profile.

The name and version information for a shared Java EE library are specified in the `META-INF/MANIFEST.MF` file.

To assemble a shared library:

1. Specify attributes for the shared library in a manifest file.
   a. Create or edit the manifest file in a text editor.

b. Enter the following commands.

```
cd <ess_user_home>/MyAppEss/EssSharedLibrary/emacs MANIFEST.MF
```

c. Add or edit a string value to specify the name of the shared Java EE library. For example:

```
Extension-Name: oracle.ess.shared
```

`Extension-Name` specifies the name of the shared Java library. Use the value specified in the script prompt for the shared library name. Oracle Enterprise Scheduler hosting applications that reference the library must specify the exact `Extension-Name` in order to use the shared files.

As a best practice, enter the optional version information for the shared Java EE library. A sample `MANIFEST.MF` file is shown in Example 14–7.

Example 14–7 Sample MANIFEST.MF File

```
Extension-Name: oracle.ess.shared
Specification-Version: 11.1.0
Implementation-Version: 11.1.0.0.0
```

d. Save the file. The `MANIFEST` file is used by the JAR deployment file.

2. Compile the project. In the Application Navigator, right-click the Oracle Enterprise Scheduler shared library project and select **Make EssSharedLibrary*.jpr***.
3. Right-click the Oracle Enterprise Scheduler shared library project and select **Project Properties** to display the Project Properties window.
4. In the Project Properties window, select **Deployment**.
5. In the Deployment Profiles region, select **EssSharedLibrary (Shared Library JAR File)**.
6. Click **Edit** to open the Edit JAR Deployment Profile Properties window.
7. In the Edit JAR Deployment Profile Properties window, click **JAR Options**.
8. In the Jar Options window, select the checkbox **Include Manifest File (META-INF/MANIFEST.MF)**.
9. Click **Add** to specify the manifest file you created. This file should be merged into the manifest file that is generated by JDeveloper.
10. In the Edit JAR Deployment Profile Properties window, expand **File Groups** and select **Filters**. Under the **Merged Contents of this File Group's Contributors** list, uncheck **essmeta**.
11. In the JAR Deployment Profile Properties page, click **OK**.
12. In the Project Properties page, click **OK**.

Task: Assemble the Hosting Application

Assembling the hosting application involves the following main steps:

- Creating a MAR deployment file.
- Updating the EAR deployment file.

To assemble the hosting application:

1. Open the Oracle Enterprise Scheduler hosting application in JDeveloper, and from the Application menu, select **Application Properties**.

2. In the Application Properties window, select **Deployment**.
3. Click **New** to display the **Create Deployment Profile** page and do the following.
   a. In the Archive Type field, from the dropdown list select **MAR File**.
   b. In the Name field enter a name, for example **myAppEss_MAR**.
   c. Click **OK**.
4. In the Edit MAR Deployment Profile Properties window, select **MAR Options**.
5. Modify the name of the MAR file, removing `_MAR` from the end of the name, for example, changing `myAppEss_MAR.mar` to `myAppEss.mar`.
6. Select the Oracle Enterprise Scheduler metadata.
   a. In the Edit MAR Deployment Profile Properties window, expand **Metadata File Groups** and select **User Metadata**.
   b. In the Order of Contributors panel on the right, click **Add** to display the Add Contributor dialog.
   c. In the Add Contributor dialog, browse to the location of the project directory, and expand it to add the `essmeta` metadata that contains the namespace for the jobs directory. Select the path that you need to include in the Add Contributor dialog by double-clicking the essmeta directory.
   d. In the Add Contributor dialog, click **OK**.
7. Select the directory for the metadata.
   a. In the Edit MAR Deployment Profile Properties window, expand Metadata File Groups and User Metadata, and select **Directories**.
   b. Select the directory that contains the Oracle Enterprise Scheduler application user metadata for the hosting application.
   c. Select the bottom most directory in the tree. This is the directory from which the namespace is created. The folder you select in this dialog determines the top level namespace in `adf-config.xml`.
   d. This namespace should be the same as the package defined in the job definition, for example `oracle/apps/ess/custom/<directory name>`.

      Note: In general, to create the namespace `oracle/apps/<product>/<component>/ess`, select the `ess` directory.

   e. In the Edit MAR Deployment Profile Properties page, click **OK**.
8. In the Application Properties window, select **Deployment**.
9. In the Deployment Profiles pane on the right, select the EAR profile and click **Edit**.
10. In the Edit EAR Deployment Profile Properties window, select **Application Assembly**.
11. Under Java EE Modules, select the checkbox for the MAR module.
12. In the Edit EAR Deployment Profile Properties window, select **EAR Options**.
13. Uncheck **Include Manifest File (META-INF/MANIFEST.MF)**.
14. In the Edit EAR Deployment Profile Properties page, click **OK**.

15. In the Application Properties page, click **OK**.

Task: Assemble the ADF Producer Application

Assembling the ADF application involves the following main steps:

- Creating an ADF library job deployment file.
- Preparing a WAR deployment profile.

Oracle ADF libraries have the option of automatic compilation that happens with deployment profile dependencies. This option allows the Oracle Enterprise Scheduler ADF library used by the user interface project to be automatically included in the `WEB-INF/lib` directory in the WAR file.

To assemble the ADF producer application:

1. Open the Oracle Enterprise Scheduler ADF application in JDeveloper.
2. In the Application Navigator, right-click the **EssModel project**. and click **New** to display the New Gallery window.
3. In the New Gallery in the Categories area, expand **General** and select **Deployment Profiles**. Create the deployment profile as follows.
   a. In the Items region, select **ADF Library Jar File**.
   b. Click **OK** to open the Create Deployment Profile window.
   c. In the Create Deployment Profile - ADF Library Jar File window, enter a name for the profile, using the format `Adf<projName>` in accordance with package structure and naming standards.
   d. Click **OK** to save the new deployment profile and close the Create Deployment Profile window.
4. In the Application Navigator, right-click the SuperWeb project and select **Project Properties**, then select **Deployment**.
5. In the Deployment Profiles region, edit the SuperWeb WAR deployment profile.
6. In the Edit WAR Profile Deployment Properties window, select **Profile Dependencies**.
7. In the pane on the right, under Java EE Modules, select the dependency under the ADF library JAR deployment file (EssModel.jpr), for example, **ADFMyApp**.
8. Click **OK** to save the WAR deployment profile.

14.3.5 Deploying Oracle Enterprise Scheduler Oracle Fusion Applications

Deploying Oracle Enterprise Scheduler Oracle Fusion applications involves the following main steps. Make sure to deploy the Oracle Enterprise Scheduler Oracle Fusion application in the order specified.

1. Deploying the shared Oracle Enterprise Scheduler library using JDeveloper or an ANT script.
2. Deploying the Oracle Enterprise Scheduler hosting application using JDeveloper or an ANT script.
3. Deploying the Oracle Enterprise Scheduler ADF producer application using JDeveloper or an ANT script.

Application-specific policies packed with script-generated hosting and ADF applications automatically migrate to the policy store when the application is

deployed. Prior to deployment, verify that any grant in the application `jazn-data.xml` file contains no duplicate permissions.

For more information about securely deploying applications, see the chapter "Deploying Secure Applications" in the *Oracle Fusion Middleware Application Security Guide*.

Task: Deploy the Shared Oracle Enterprise Scheduler Library Using JDeveloper

You can deploy the shared Oracle Enterprise Scheduler library using JDeveloper or an ANT script.

To deploy the share library using JDeveloper:

1. In the Application Navigator, right-click the Oracle Enterprise Scheduler shared library project, select **Deploy** and then select the shared library JAR.

   The Deploy EssSharedLibrary_JAR window displays.

2. Select **Deploy to a Weblogic Application Server** and click **Next**.
3. In the Select Server window, select the application server to which you want to deploy the Oracle Enterprise Scheduler shared library.
4. Click the **Add** button to create a connection to the application server if none is defined. Click **Next**.
5. In the Weblogic Options window, make the following selections:
   a. Select **Deploy to selected instances in the Domain**, and choose the Oracle Enterprise Scheduler server instance in the table row. The Oracle Enterprise Scheduler shared library should be deployed to the same server as the Oracle Enterprise Scheduler hosting application.
   b. Select **Deploy as a shared library**.
   c. Click **Finish**.
6. Verify the deployment using the deployment log. Upon successful deployment, you can see the Oracle Enterprise Scheduler jobs shared library deployed as 'oracle.ess.shared(11,11.1.1)' in the Oracle Weblogic Administration Console.

Task: Deploy the Shared Oracle Enterprise Scheduler Library Using an ANT Script

To deploy the shared library using an ANT script:

1. Run the following ant command:

   ```
   ant -f ${ESS_HOME}/ant/build-ess.xml deploy_job_logic
   ```

   where `deploy_job_logic` builds, packages and deploys only the Oracle Enterprise Scheduler jobs shared library.

   Note: When prompted, enter the Oracle WebLogic Server password.

2. To specify a different value for the ESS shared library name, take the following steps:
   a. In a text editor, modify the shared library JAR `MANIFEST` file. For example:

   ```
   vi ${ess_user_home_dir}/MyAppEss/EssSharedLibrary/MANIFEST.MF
   ```

b. Edit the string value of `Extension-Name` to specify the name of the shared Java EE library.

c. Enter the optional version information for the shared Java EE library.

d. Update the Oracle Enterprise Scheduler `build.properties` file by editing `${ESS_HOME}/ant/config/ess-build.properties`.

e. Change the value of the property `customEss.shared.library.name` to match the value specified in the JAR `MANIFEST` file. A sample build.properties file is shown in Example 14–8.

Example 14–8 Sample build.properties file

```
# ESS build properties
 ess.script.base.dir=${user_home}

 fmw.home=${fmw_home}
 jdev.home=${fmw.home}/jdeveloper
 oracle.common=${fmw.home}/oracle_common

 # ========== ESS JDev project details ===============
 customEss.project.dir=${ess.script.base.dir}
 customEss.hostapp.workspace=${hosting_application_name}
 customEss.hostapp.jwsfile=${hosting_application_name}
 customEss.hostapp.earprofile=${hosting_application_name}
 customEss.hostapp.jprproject=EssSharedLibrary
 customEss.hostapp.jarprofile=EssSharedLibrary
 customEss.hostapp.jarfile=${jobdef_library_name}

 customEss.shared.library.name=${jobdef_library_name}

 customEss.hostapp.mds.partition=globalEss
 customEss.hostapp.mds.jdbc=mds-ApplicationMDSDB
 customEss.hostapp.name=${hosting_application_name}

 customEss.producerapp.workspace=${ui_application_name}
 customEss.producerapp.jwsfile=${ui_application_name}
 customEss.producerapp.earprofile=${ui_application_name}
 customEss.producerapp.name=${ui_application_name}

 # ========== Weblogic Server details ===============
 MW_HOME=${fmw.home}
 ORACLE_HOME=${jdev.home}
 MW_ORA_HOME=${jdev.home}
 COMMON_COMPONENTS_HOME=${oracle.common}
 WEBLOGIC_HOME=${fmw.home}/wlserver_10.3
 weblogic.server.host=<server_host>
 weblogic.server.port=<server_port>

 weblogic.server.ssl.port=<server_ssl_port>

 weblogic.admin.user=<admin_username>
 weblogic.t3.url=t3://${weblogic.server.host}:${weblogic.server.port}
 # WLS server name where ESS producer web app is targeted for deployment
 adfapp.server.name=AdminServer
 # WLS server name where ESS hosting app is targeted for deployment
 ess.server.name=ess_server1
```

f. Save the file.

Task: Deploy the Oracle Enterprise Scheduler Hosting Application Using JDeveloper

You can deploy the Oracle Enterprise Scheduler application using JDeveloper or an ANT script.

To deploy the Oracle Enterprise Scheduler hosting application using JDeveloper:

1. In JDeveloper, open the Oracle Enterprise Scheduler hosting application.
2. From the Application menu, select **Deploy** and then select the name of the hosting application, for example **MyAppEss**.
3. In the Deploy MyAppEss window, select **Deploy to Application Server** and click **Next**.
4. In the Select Server window, select the application server to which you want to deploy the Oracle Enterprise Scheduler hosting application.

   Click the Add button to create a connection to the application server if none is defined.
5. Click **Next**. In the Weblogic Options window, make the following selections:
   a. Select **Deploy to selected instances in the Domain**, and choose the Oracle Enterprise Scheduler server instance in the table row, to which the Oracle Enterprise Scheduler hosting application is to be deployed.
   b. Select **Deploy as a standalone Application**.
   c. Click **Finish**.

   JDeveloper displays the Deployment Configuration page. Select the relevant options for your metadata repository.
6. Click **Deploy**.

   Verify the deployment using the deployment log.

   Upon successful deployment, you can expect to see the Oracle Enterprise Scheduler hosting application deployed in Fusion Applications Control.

Task: Deploy the Oracle Enterprise Scheduler Hosting Application Using an ANT Script

To deploy the Oracle Enterprise Scheduler using an ANT script:

- Run the following ANT command:

  ```
  ant -f ${ESS_HOME}/ant/build-ess.xml deploy_ess_host
  ```

  where `deploy_ess_host` builds, packages and deploys only the Oracle Enterprise Scheduler hosting application. It is assumed that the Oracle Enterprise Scheduler shared job library is already deployed prior to running this command.

> **Note:** When prompted, enter the Oracle WebLogic Server password.

Task: Deploy the ADF Producer Application Using JDeveloper

You can deploy the ADF producer application using Oracle JDeveloper or an ANT script. This step is optional if using an existing deployed producer web application. The value you defined for **EXT_PortletContainerWebModule** in Section 14.3.3.1 indicates the name of the application to be used.

To deploy the ADF producer application using JDeveloper:

1. In JDeveloper, open the ADF producer application.
2. Click the Application menu, select **Deploy** and then select the name of the ADF producer application.
3. In the Deploy MyApp window, select **Deploy to Application Server** and click **Next**.
4. In the Select Server window, select the application server to which you want to deploy the Oracle Enterprise Scheduler ADF application.
5. Click the Add button to create a connection to the application server if none is defined.
6. Click **Next**. In the Weblogic Options window, make the following selections:
   a. Select Deploy to selected instances in the Domain, and choose the Oracle Enterprise Scheduler server instance in the table row, to which the Oracle Enterprise Scheduler ADF application is to be deployed.
   b. Select **Deploy as a standalone Application**.
   c. Click **Finish**.
   d. The Select Deployment Type dialog window displays, prompting you to expose the MyApp portlet application as a WSRP service. Select **Yes**.
7. Click **Next**. The Deployment Configuration displays. Select the relevant options for your metadata repository.
8. Enter **globalEss** as the partition name.
9. Click **Deploy**.
10. Verify the deployment using the deployment log.

    Upon successful deployment, you can expect to see the deployed Oracle Enterprise Scheduler ADF application in Fusion Applications Control.
11. Launch the WSRP Producer test page to validate the deployment using the following URL:

    ```
    http://<ADF_HOST>:<ADF_PORT>/<MyApp-context-root>/
    ```

Task: Deploy the ADF Producer Application Using an ANT Script

To deploy the ADF producer application using an ANT script:

- Run the following ANT command:

  ```
  ant -f ${ess_user_home_dir}/ant/build-ess.xml deploy_ess_ui
  ```

  where `deploy_ess_ui` builds, packages and deploys only the Oracle Enterprise Scheduler producer ADF application.

> **Note:** When prompted, enter the Oracle WebLogic Server password.

14.3.6 Registering Oracle Enterprise Scheduler Topology Objects

Registering Oracle Enterprise Scheduler topology objects involves the following main steps:

- Creating Oracle Enterprise Scheduler topology objects,
- Registering Oracle Enterprise Scheduler topology objects.

> **Note:** You need only register the topology objects when using an ANT script generated ADF producer web application. Alternatively, you can use an existing registered web or Oracle Enterprise Scheduler ADF producer application and skip this section.

Task: Create Oracle Enterprise Scheduler Topology Objects

Use the Setup and Maintenance work area to create Oracle Enterprise Scheduler topology objects, including the Oracle Enterprise Scheduler domain, hosting application and Oracle Enterprise Scheduler producer Oracle ADF application.

To create Oracle Enterprise Scheduler topology objects:

1. Create the Oracle Enterprise Scheduler domain topology object.
   a. In the global area of Oracle Fusion Applications, select the **Administration** menu and then select the **Setup and Maintenance** work area.
   b. From the Tasks Pane, select **Topology Objects** and then select **Manage Domains**.
   c. On the Manage Domains page in the list of domains, click the **Actions** drop-down list and select **Create**.
   d. In the Create Domain window that displays, enter a name for the domain and click **Save and Close**.
2. Create the Oracle Enterprise Scheduler hosting application topology object.
   a. In the global area of Oracle Fusion Applications, select the **Administration** menu and then select the **Setup and Maintenance** work area.
   b. From the Tasks Pane, select **Topology Objects** and then select **Manage Enterprise Applications**.
   c. On the Manage Enterprise Applications page in the list of domains, click the **Actions** drop-down list and select **Create**.
   d. In the Create Enterprise Application page, enter the following details.

Table 14–4 Enterprise Application Topology Object Details

Field	Description
Name	Enter the name of the enterprise application that you want to register, for example **EarCustomHostEss**.
Code	Enter a unique code to identify the enterprise application. Once you have created it, the code cannot be changed.
Domain	Select the name of the domain to be used by the enterprise application, for example **EssDomain**.
Default URL	Enter a static URL if the enterprise application is always to be deployed at the same location. Optional.
Source File	Enter the name of the EAR file. Optional.
Pillar	From the Available Pillars list, shuttle the relevant pillar or pillars to the Selected Pillars list.

e. Click **Save and Close** to create the Oracle Enterprise Scheduler hosting application topology object.

3. Repeat the previous step to create the Oracle Enterprise Scheduler producer Oracle ADF application topology object.

Task: Register Oracle Enterprise Scheduler Topology Objects

Use the Setup and Maintenance work area to register the Oracle Enterprise Scheduler topology objects you created.

To register Oracle Enterprise Scheduler topology objects:

1. Register the Oracle Enterprise Scheduler domain.
   a. In the global area of Oracle Fusion Applications, select the **Administration** menu and then select the **Setup and Maintenance** work area.
   b. From the Tasks Pane, select **Topology Registrations** and then select **Register Domains**.
   c. On the Register Domains page in the list of domains, click the **Actions** drop-down list and select **Create**.
   d. In the Add Domain window that displays, enter the details for the Oracle Enterprise Scheduler domain created in "Task: Create Oracle Enterprise Scheduler Topology Objects".

Table 14–5 Domain Registration Values

Field	Description
Enterprise Environment	From the dropdown list, select the enterprise environment to be used, for example `oracle`.
Domain	From the dropdown list, select the name of the domain.
Name/Administrator Server Name	Enter a name for the registered domain. Enter a name for the domain's administrator server.
Internal/External/Administrator Server Host/Port/Protocol	Enter the URL, port number and protocol (such as HTTP, HTTPS, and so on) for the internal server to be registered, as well as the externally-facing server and the administrative server.
Enterprise Manager Protocol	From the dropdown list, select the protocol to be used for accessing Oracle Enterprise Manager, for example HTTP or HTTPS.
Enterprise Manager Port	Enter the port number to be used when accessing Oracle Enterprise Manager in the domain.
Java Management Extensions Port	Enter the port number to be used for Java management extensions.

   e. Click **Save and Close** to save your changes.

2. Register the Oracle Enterprise Scheduler web producer module.
   a. In the global area of Oracle Fusion Applications, select the **Administration** menu and then select the **Setup and Maintenance** work area.
   b. From the Tasks Pane, select **Topology Objects** and then select **Manage Modules**.
   c. On the Manage Modules page from the list of applications, click the **Actions** drop-down list and select **Register Modules**.

d. In the Register Modules window that displays, enter the following details.

Table 14–6 Domain Registration Values

Field	Description
Name	Enter the name of the module that you want to register.
Code	Enter a unique code to identify the module. Once created, the code cannot be changed.
Description	Enter a brief, meaningful description of the module. Optional.
Enterprise Application	Select and associate the enterprise application to which the module belongs.
Type	Select the relevant module type from the list.
Context Root	Enter the context root of the module.

e. Click **Save and Close** to save your changes.

3. Register the Oracle Enterprise Scheduler hosting and producing applications.

a. In the global area of Oracle Fusion Applications, select the **Administration** menu and then select the **Setup and Maintenance** work area.

b. From the Tasks Pane, select **Topology Registrations** and then select **Register Enterprise Applications**.

c. On the Register Enterprise Applications page from the list of applications, click the **Actions** drop-down list and select **Add**.

d. In the Add Enterprise Application window that displays, enter the following details.

Table 14–7 Domain Registration Values

Field	Description
Enterprise Environment	From the dropdown list, select the enterprise environment to be used, for example `oracle`.
Enterprise Application	Select and associate the enterprise application to which the module belongs.
Name	Enter the name of the enterprise application.
External Server Protocol/Host/Port	Enter the URL, port number and protocol (such as HTTP, HTTPS, and so on) for the external server to be registered with the enterprise application.

e. Click **Save and Close** to save your changes.

f. In the Register Enterprise Applications page, click the **Actions** drop-down list and select **Add** to display the Add Enterprise Application window.

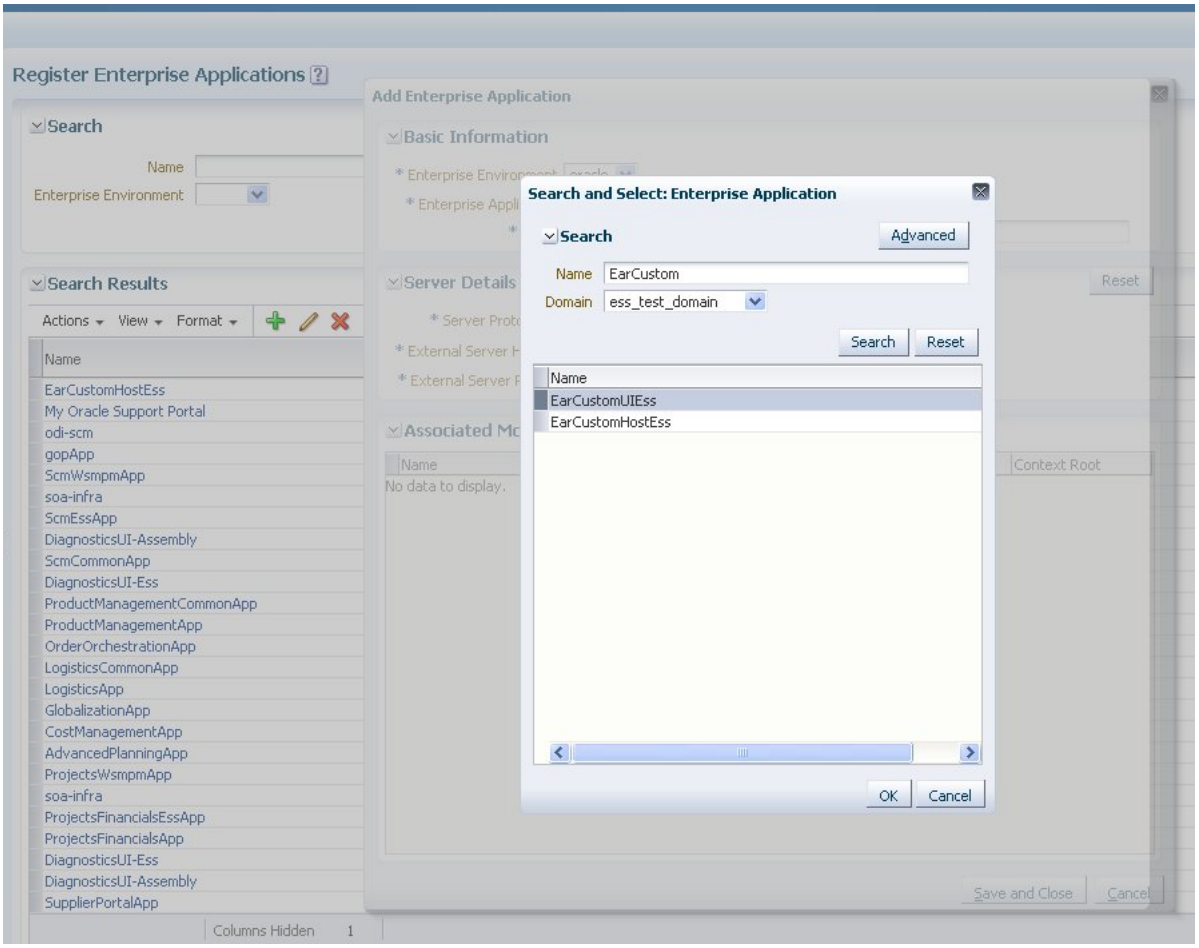
g. Click the Enterprise Application dropdown list to display the Search and Select: Enterprise Application window.

h. In the **Name** field, enter a name for the application you want to search for and click the **Domain** dropdown list to select the domain in which you want to search.

Click **Search** to search for the Oracle Enterprise Scheduler producer web application.

i. From the list of enterprise applications that displays, select the relevant Oracle Enterprise Scheduler producer web application and click **OK**, as shown in Figure 14–2.

Figure 14–2 Select the relevant enterprise application



j. In the Add Enterprise Application page, fill in the details for the Oracle Enterprise Scheduler producer web application as described in Table 14–7.

k. Click **Save and Close**.

14.3.7 Granting Job Metadata Permissions to Application Roles and Users

You can use Oracle Authorization Policy Manager to manage application roles and resource-based policies. Identifying the application roles and users, and granting them the required privileges to execute Oracle Enterprise Scheduler job-related tasks is a one time operation.

Granting Oracle Enterprise Scheduler metadata permission to the new job involves the following main steps:

- Creating a new resource for the custom job definition,
- Creating a new policy.

Task: Create a Resource

In Oracle Authorization Policy Manager, create an application resource instance.

To create a resource:

1. Run Oracle Authorization Policy Manager by entering the following URL in a browser.

   ```
   http://<fs-domain_url>/apm/
   ```

2. From the navigation pane, right-click the application Resources icon and select **New**.

   An untitled page displays.

3. Define a resource with the resource type **ESSMetadataResourceType**, as well as the name and display name of the Oracle Enterprise Scheduler component using the following syntax:
   `oracle.apps.ess.`*`applicationName`*`.`*`JobDefintitionName`*`.`*`JobName`*.

4. Save the resource.

Task: Define a Policy

Define a policy that specifies the privileges allocated to a particular user when submitting the job request.

To define a policy:

1. In Oracle Authorization Policy Manager in the Home tab, under the Applications region, choose an application for which you want to manage the policy (for example, **MyAppEss**).
2. Click **Search Policies** to display the Search Authorization Policies tab.
3. In the Search Authorization Policies tab, select the principal user on which to base the policy being created, for example, **FinUser1**.
4. In the Functional Security tab, select **Resource Based Policies**.
5. Click **New Policy** to create a new policy for the selected user.
6. Add resource instances to the policy by clicking the **Add** button in the Resources table.
7. Select the resource instance created for the custom Oracle Enterprise Scheduler job definition (from the previous task).
8. Specify the actions `EXECUTE` and `READ` to provision Oracle Enterprise Scheduler job execution privileges to the user.
9. Click **Save**.

Task: Test Oracle Enterprise Scheduler Job Submission from the Oracle Enterprise Scheduler Central UI

Submit a job request to make sure everything works as it should.

To submit a test job request:

1. Login to Functional Setup Manager with the user for whom you defined an authorization policy (for example, as **FinUser1**).

   The URL for Functional Setup Manager is as follows.

   ```
   https://<HOST>/setup/faces/TaskListManagerTop
   ```

2. From the **Tools** menu, select **Schedule Processes**.

3. Click the **Schedule New Process** button and select a job process name when prompted. Select the job definition you created.
4. Click **OK**.

   The Oracle Enterprise Scheduler Schedule Request Submission window displays.
5. In the Parameters region, specify the job parameters as required.
6. Click **Submit** to schedule the job execution, and **Close** to exit the window.
7. Refresh the Search Results table to monitor the status of submitted job.

14.4 Customizing Existing Oracle Enterprise Scheduler Job Properties

You can customize Oracle Enterprise Scheduler jobs that are associated with Oracle Fusion applications. Customizing existing Oracle Enterprise Scheduler jobs involves editing job properties using Oracle Enterprise Manager Fusion Applications Control.

An example of a customization is to set the time-out value for a scheduled job to be run asynchronously. When the job takes longer than the time-out, you can find the job that timed out in Fusion Applications Control and manually complete the job.

The job properties that can be edited are shown in Table 14–8.

For more information about editing scheduled job properties, see the chapter "Managing Oracle Enterprise Scheduler Service and Jobs" in *Oracle Fusion Applications Administrator's Guide*.

Table 14–8 Job Properties

API	Description
`oracle.as.scheduler.SystemProperty.PRIORITY`	This property specifies the request processing priority, from 0 to 9, where 0 is the lowest priority and 9 is the highest. If this property is not specified, the system default value used is `oracle.as.scheduler.RuntimeService#DEFAULT_PRIORITY`.
`oracle.as.scheduler.SystemProperty.RETRIES`	This property defines the numerical value that specifies the retry limit for a failed job request. If job execution fails, the request retries up to the number of times specified by this property until the job succeeds. If the retry limit is zero, a failed request will not be retried. If this property is not specified, the system default used is `oracle.as.scheduler.RuntimeService#DEFAULT_RETRIES`.
`oracle.as.scheduler.SystemProperty.REQUEST_CATEGORY`	This property specifies an application-specific label for a request. The label, defined by an application or system administrator, allows administrators to group job requests according to their own specific needs.

Table 14–8 (Cont.) Job Properties

API	Description
`oracle.as.scheduler.SystemProperty.ASYNC_REQUEST_TIMEOUT`	This property specifies the time in minutes that the job request processor waits for an asynchronous request after it has begun execution. After the time elapses, the job request times out.
`enableTrace`	The property specifies a numerical value that indicates the level of tracing control for the job. Possible values are as follows: ■ 1: Database trace ■ 5: Database trace with bind ■ 9: Database trace with wait ■ 13: Database trace with bind and wait ■ 16: PL/SQL profile ■ 17: Database trace and PL/SQL profile ■ 21: Database trace with bind and PL/SQL profile ■ 25: Database trace with wait and PL/SQL profile ■ 29: Database trace with bind, wait and PL/SQL profile
`enableTimeStatistics`	This property enables or disables the accumulation of time statistics.

15

Customizing Security for ADF Application Artifacts

This chapter describes how to customize security for custom and extended business objects and related custom and extended application artifacts defined by Oracle Application Development Framework (Oracle ADF) in Oracle Fusion applications using Oracle Authorization Policy Manager and Oracle JDeveloper.

Security customization in the production environment is typically restricted to the security administrator using Oracle Authorization Policy Manager; however, during the development phase of application customization, you can perform similar security customization tasks using Oracle Authorization Policy Manager and JDeveloper.

This chapter includes the following sections:

15.1 About the Oracle Fusion Security Approach

Oracle Fusion Applications is secure as delivered. The Oracle Fusion security approach tightly coordinates various security concerns of the enterprise, including:

- The ability to define security policies to specify the allowed operations on application resources, including viewing and editing data and invoking functions of the application.
- The ability to enforce security policies by roles assigned to end users, and not directly enforced on the end users of the system.

For example, when an end user attempts to access a task flow, whether or not the end user has the right to enter the flow and view the contained web pages is specified by the roles provisioned to the end user and the security policies defined for those roles.

In the enterprise, the security administrator ensures end users are provisioned with sufficient access rights to perform the duties of their various jobs. These provisioning tasks involve Oracle Fusion Middleware tools that integrate with Oracle Fusion Applications and allow IT personnel to extend the *Oracle Fusion security reference*

implementation, which consists of predefined security policies and roles used in Oracle Fusion Applications. These tools directly update a copy of the reference implementation in the deployed application's security policy store and identity store.

From the standpoint of the application developer who seeks to apply the Oracle Fusion security approach to an Oracle Fusion application that they extend, the security implementation process overlaps with tasks performed by IT personnel. You may or may not need to extend the Oracle Fusion security reference implementation, depending upon how end users will interact with the new resource. At the end of the process, you must ensure that any new resource you create, such as a business object in the data model project or a task flow in the user interface project, has sufficient security policies to grant access privileges and suitable roles to receive the access privileges.

15.1.1 How to Proceed With This Chapter

Customizing security is a complex process that involves working with several tools, familiarity with diverse technologies, and coordination between the application developer and security administrator. For a concise summary of the security customization scenarios and corresponding tasks, see Table 15–1 in Section 15.3.3, "Oracle Fusion Security Customization Scenarios."

After familiarizing yourself with the types of security customizations performed by the application developer, read the following sections to gather a more complete understanding of the security customization process.

- For an overview of the Oracle Fusion security reference implementation, see Section 15.2.
- For a list of Oracle Fusion security guidelines that dictate which security artifacts in the Oracle Fusion security reference implementation you may or may not modify, see Section 15.3.1.
- For an overview of the steps you follow to secure a new resource, see Section 15.3.2.
- For additional background about the type of resource customizations that require customizing security, see Section 15.3.4 and Section 15.3.5.
- For details about the security artifacts that you define to create security policies, see Section 15.3.6 through Section 15.3.9.
- For a list of prerequisite tasks to be completed before customizing security, see Section 15.3.11.
- For information about the tools involved in customizing security, see Section 15.4 through Section 15.6.

15.1.2 Related Security Documents

The following related documents contain important information specific to customizing security in Oracle Fusion Applications. References to these documents appear throughout this chapter. Please consult these documents for complete details.

- *Oracle Fusion Applications Security Guide*

  Describes the concepts and best practices of the Oracle Fusion security approach. This is the main document addressing the Oracle Fusion security approach.

- *Oracle Fusion Applications Security Hardening Guide*

Describes how the security administrator proceeds to implement the Oracle Fusion security reference implementation for their enterprise.

- *Oracle Fusion Applications security reference manuals*

  Describes the segregation of duties in the Oracle Fusion security reference implementation. Each Oracle Fusion application has its own reference manual.

- *Oracle Fusion Applications Developer's Guide*

  Describes how to secure new custom resources in Oracle Fusion Applications. Includes chapters describing how to implement data security and function security for new resources.

- *Oracle Fusion Applications Administrator's Guide*

  Summarizes available security administration tasks in a single chapter.

- *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*

  Describes how to create and modify data security policies and data role templates.

- *Oracle Fusion Middleware Application Security Guide*

  Describes the concepts and best practices of Oracle Platform Security Services (OPSS) upon which Oracle Fusion security is based. This is the main document addressing the architecture of Oracle security services.

- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

  Describes ADF Security, through which Oracle ADF components interact with OPSS.

- *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*

  Describes role provisioning and other identity management tasks.

- *Oracle Database Security Guide*

  Describes implementing security policies at the level of the database.

- JDeveloper online help topics

  Describes the tools used to create database objects using JDeveloper.

15.2 About Extending the Oracle Fusion Security Reference Implementation

The Oracle Fusion security approach is embodied in the Oracle Fusion security reference implementation which delivers predefined roles and security policies that address the common business needs of the enterprise. The reference implementation can be extended to adjust to the needs of a specific enterprise. The predefined security policies implement role-based access control: a set of roles recognizable as jobs, a role hierarchy that contains the duties for those jobs, and a set of role provisioning events and workflows. The Oracle Fusion security reference implementation represents the security guidelines of what Oracle considers to be the general case for jobs, roles, duties, and segregation of duties.

In general, the Oracle Fusion security reference implementation is designed to require only small changes to adjust Oracle Fusion security for a specific enterprise. The reference implementation provides a comprehensive set of predefined security policies and predetermined data role templates that may be customized to generate security

policies. From the standpoint of the security administrator who addresses the specific security concerns of their organization, typical tasks include changing or extending role definitions and role hierarchies and managing security policies and data role templates. For example, enterprise IT security personnel eventually review the duties and access defined in the reference implementation and specify how that matches with the job titles and tasks the enterprise expects to be performed in the deployed Oracle Fusion application.

Provisioning of end users with role membership is defined in the application's identity store and is a configuration task to be performed by the security administrator independent of security customization. The reference implementation contains four types of roles: abstract, job, duty, and data roles and implements hierarchies between these roles to streamline provisioning access to end users. Each of the Oracle Fusion Applications roles is implemented in Oracle Fusion Middleware as one of these roles:

- Internal roles are roles that are not assigned directly to end users, also called *application roles* because they are specific to an application.

  Note that, in Oracle Fusion Applications, application roles are called *duty roles*. The Oracle Fusion security reference implementation defines a large number of duty roles that correspond to the duties of individual job roles. Duty roles are specific to applications, stored in the policy store, and shared within an Oracle Fusion Applications instance.

- External roles are roles associated with a collection of end users and other groups, also called *enterprise roles* because they are shared across the enterprise.

  In Oracle Fusion Applications, enterprise roles include *job roles*, *data roles*, and *abstract roles*.

The *job role* is a role that corresponds to a job title defined in Human Resources (HR). The *duty role* is a role that corresponds to a line on a job description for that job. For example, in your enterprise, the job of an `Application Developer` may also include `Project Management Duties`. The *data role* is a role that authorizes a person with a job to a particular dimension of data on which they can work: for example, `AP Manager - US Commercial Business Unit` identifies who may access the accounts specific to the US division of the enterprise. The *abstract role* is a role that is not a job title, but is a means to group end users without respect to specific jobs: for example, `Employee` or `Line Manager`.

The division between internal roles and external roles is an important principle of the Oracle Fusion security approach. The principle, called *least privilege*, ensures that the end user only acquires privileges specific to the job they perform rather than a variety of miscellaneous duties. Therefore, in adherence of the principle of least privilege, duty roles are defined by Oracle Platform Security Services (OPSS) as internal roles and cannot be assigned directly to end users.

To understand the Oracle Fusion security approach in detail and to learn more about using the Oracle Fusion security infrastructure to implement and administer security for the enterprise, see the "Introduction" chapter in the *Oracle Fusion Applications Security Guide*.

15.3 About Extending and Securing Oracle Fusion Applications

Oracle Fusion Applications is configured by default to deny end users access to the data of the application domain and the web pages that display the data. An important principle of Oracle Fusion security ensures that end users do not have unintended access to data and application artifacts exposed in the extended application.

To enable access to custom resources in the extended application, you may create security policies to specify "who can perform what operations on what specific data and on what specific application artifacts."

> **Note:** The term *protected* in this chapter refers to the default Oracle Fusion Applications condition that denies end users access to database resources and application artifacts. In contrast, the term *secured* refers to resources that have been made accessible to end users through security policies created for this purpose. Therefore, a security policy specifically *enables access* to the resource based on the access rights it confers to the end user.

To create the security policy, you must consider the additional duties the end users of the extended application will perform and then grant the desired roles specific rights to:

- Access the web pages of a custom task flow that supports the duty
- Access the specific data records, or instances of a custom business object, required to complete the duty
- Perform only those operations on that data required by the duty

When you need to secure new resources, you can expect to work with two different types of security policies: *data security policies* that control access to the data records of database tables or views in the Oracle Fusion Applications schema and *function security policies* that control access to the Oracle Fusion application artifacts used to display the data. Because the representation of data security policies and function security policies differ, the environment you will use to define security policies depends on whether data security or function security is being implemented.

In the case of access to data records, a custom business object may be secured either implicitly or implicitly. For example, the AP Manager is authorized to an explicit list of business units specified by a data role, whereas the Project Manager is implicitly authorized to the projects that he manages. When you need to secure data records, then you can:

- Implicitly grant data access to abstract and job roles through data security policies you define on custom duty roles inherited by the abstract or job role.

  You can create custom duty roles to support a new duty introduced by a custom application resource.

- Explicitly grant data access to a data role through a data security policy you apply directly to the inherited job or abstract role through a data role template.

  You can customize the data role template before running the template to generate the data roles.

15.3.1 Oracle Fusion Security Customization Guidelines for New Functionality

In general, when you create new functionality, not supported by Oracle Fusion Applications, do not include authorization to that functionality from within the security artifacts that Oracle Fusion Applications delivers in the reference implementation.

Specifically, Oracle Fusion security guidelines suggest customization developers and security administrators *must not modify* the following security artifacts in the reference

implementation when introducing *new functionality*, through custom or extended business objects.

- Predefined duty roles, specifically:
  - Do not change the role hierarchy by removing member duty roles assigned to parent duty roles or job roles.
  - Do not remove (also called *revoke*) existing privileges granted to duty roles.
  - Do not add (also called *grant*) new privileges to duty roles.
- Predefined security policies (including data and function), specifically:
  - Do not remove existing instance sets from predefined data security policies.
  - Do not remove existing member resources from predefined function security policies.
  - Do not revoke existing actions (mapped by Oracle Fusion security to resource operations) granted on each resource or instance set.

Customization developers and security administrators *may modify* security artifacts in the reference implementation in the following ways.

- Do modify job roles to add a custom duty role (permissible by security administrator only).
- Do modify data role templates to add a new job role as the base role or to add access privileges to a custom business object.

Customization developers and security administrators *may create* the following security artifacts and add them to the reference implementation.

- Do create custom duty roles when a custom application resource requires a new duty role to support the segregation of duties or when a custom application resource introduces new privileges to a predefined business object.
- Do create data role templates when a custom business object is used as a data stripe and explicit data security policies grant access to the data stripe. For example, when data roles grant access to a specific business unit or organization.

> **Note:** You must not modify predefined duty roles, and you must always add custom duty roles to grant access rights. Only the security administrator can add or remove duty roles associated with an existing job role. If a predefined job role does not exist that adequately describes the duties performed by a job, then the security administrator can also create a new job role.

15.3.2 Oracle Fusion Security Customization Process Overview

Creating a new, custom business object and exposing it in the extended application is one of the main customization tasks that you may perform. Although you may also extend existing business objects to introduce new functionality or to introduce additional data, the security customization process for new and existing business objects follows a similar pattern.

To security a *new* business object in the extended Oracle Fusion application:

1. Create a custom duty role to serve as the grantee of the security policy privileges.

For details about creating duty roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

2. Define a database resource in the Oracle Fusion Data Security repository to protect the data records of a database table that you intend to expose in the application.

   For details about registering a database table as a database resource, see Section 15.3.6, "What You Can Customize in the Data Security Policy Store at Design Time."

   This step causes Oracle Fusion security to protect the database table records, thus rendering the data inaccessible to the end user of the application. A data security policy will be required to grant access to the data defined by the database resource and a function security policy will be required to grant access to the application artifacts that display the data in the extended application.

3. Define data security policies for the previously defined database resource to grant access to specific data records for a given role.

   For details about securing data, see Section 15.3.6, "What You Can Customize in the Data Security Policy Store at Design Time."

4. Extend the data model project (in the extended application) with a new entity object to expose the database table that you defined as an Oracle Fusion Data Security database resource.

   For details about creating custom business components to represent a database table, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

5. Opt into the previously defined data security policies by enabling OPSS authorization checking on the operations of individual data model objects in the data model project.

   For details about enabling security, see Section 15.3.7, "What You Can Customize in the Data Model Project at Design Time."

6. Consult a security administrator to export all predefined function security policies of the application that you are customizing into a `jazn-data.xml` file.

   For details about how the security manager exports the application policy store, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

7. Copy the exported `jazn-data.xml` file into your application workspace.

   For details about adding the file to your application, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

8. Customize the Oracle ADF application artifacts in the user interface project to display the data records exposed by the extended data model.

   For details about creating securable custom application artifacts, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

9. Define function security policies for the custom Oracle ADF application artifacts to specify the access rights of end users.

   For details about securing application functions, see Section 15.3.9, "What You Can Customize in the Application Security Policy Store at Design Time."

10. Opt into the previously defined function security policies by running the ADF Security wizard to enable OPSS authorization checking.

For details about enabling security on the user interface project, see Section 15.3.8, "What You Can Customize in the User Interface Project at Design Time."

15.3.3 Oracle Fusion Security Customization Scenarios

You do not need to customize security for every type of customization that you may make in the extended application. Whether or not a security policy is needed will depend on the application resource and the type of customization performed.

Table 15–1 summarizes the security customization scenarios that Oracle Fusion security supports. The "Application Developer Tasks" column of the table provides a brief description of the security artifacts involved in each scenario, but presumes some familiarity with the Oracle Fusion security approach (for guidance see Section 15.1.1, "How to Proceed With This Chapter").

Note: For simplicity, Table 15–1 does not make a distinction between explicit and implicit data security policies. You may also need to customize data role templates when a custom business object is used as a data stripe and explicit data security policies grant access to that data stripe. For more details about customizing data role templates, see Section 15.3.6, "What You Can Customize in the Data Security Policy Store at Design Time."

Table 15–1 Oracle Fusion Applications Security Customization Use Cases

Security Customization Goal	Security Policy Requirement	Application Developer Tasks
Control whether the end user associated with a particular role may access a **new task flow** and view all the web pages of the flow.	Create a new security policy. The new task flow will be inaccessible by default (also called *protected*) and will require a new function security policy to grant end users access.	Enable ADF Security on the user interface project to protect all task flows (and the web pages they contain). Then, in the file-based policy store, create a resource definition for the task flow and assign the definition as a member of an entitlement (defined in the policy store as a *permission set*) that you name. Then, create the security policy by granting the entitlement to a custom application role that you either created or consulted with a security administrator to create for you. As a security guideline, do not modify a predefined function security policy by granting additional entitlements to an predefined duty role.
Control whether the end user associated with a particular role may access a **customized task flow** and view the new or customized web pages of the flow.	Do not create a security policy. The customized Oracle Fusion application task flow will have an predefined function security policy defined by the Oracle Fusion security reference implementation; since this type of change does not require new duties, there is no need to grant access to a new duty role.	Consult the security administrator who can make a customized task flow accessible to additional end users through role provisioning. If the same group of end users require access to the customized task flow, then no change to the provisioned end users is required.

Table 15–1 (Cont.) Oracle Fusion Applications Security Customization Use Cases

Security Customization Goal	Security Policy Requirement	Application Developer Tasks
Control whether the end user associated with a particular role may access a **new top-level web page**. In Oracle Fusion Applications, a top-level web page is one that is not contained by a task flow.	Create a new security policy. The new top-level web page will be inaccessible by default (also called *protected*) and will require a new function security policy to grant end users access. The ability to secure individual web pages in Oracle Fusion Applications is reserved for top-level web pages backed by an ADF page definition file only.	Enable ADF Security on the user interface project to protect all top-level web pages backed by ADF page definition files. Then, in the file-based policy store, create a resource definition for the web page and assign the definition as a member of an entitlement (defined in the policy store as a *permission set*) that you name. Then, create the security policy by granting the entitlement to a custom application role that you either created or consulted with a security administrator to create for you. As a security guideline, do not modify a predefined function security policy by granting additional entitlements to an predefined duty role.
Control whether the end user associated with a particular role may access a **customized top-level web page**. In Oracle Fusion Applications, a top-level web page is one that is not contained by a task flow.	Do not create a security policy. The customized top-level web page will have an predefined function security policy defined by the Oracle Fusion security reference implementation; since this type of change does not require new duties, there is no need to grant access to a new duty role.	Consult the security administrator who can make customized top-level web pages accessible to additional end users through role provisioning. If the same group of end users requires access to the web page, then no change to the provisioned end users is required.
Decide whether the end user associated with a particular role has the right to select **create, edit, or delete button** in the displayed web page.	Do not create a security policy. Access to user interface components, such as buttons, is not controlled by a security policy, but can be controlled by rendering the button in the user interface based on the end user's role.	Conditionally render the component by entering ADF Security Expression Language (EL) utility methods on the `rendered` attribute of the button to test whether the end user has membership in a particular role.
Control whether the end user associated with a particular role may view or update a **specific set of data records for an all new business object** in the displayed web page.	Create a new security policy. After an Oracle Fusion Database Security database resource is defined for the data, the data records exposed by the new business object will be inaccessible by default (also called *protected*) and will require a new data security policy to grant end users read or update access on one or more specific sets of data records.	Enable authorization checking on the appropriate operations of the ADF entity object (`read`, `update`, and `removeCurrentRow`) that maps to a specific database table. Then, in the Oracle Fusion Data Security repository, add a custom duty role as the grantee of access privileges and create a named instance set of data records. Then, create the security policy by granting Oracle Fusion Data Security view or update privileges to the custom duty role for the data records. As a security guideline, do not modify a predefined data security policy by granting additional privileges to an predefined duty role.

Table 15–1 (Cont.) Oracle Fusion Applications Security Customization Use Cases

Security Customization Goal	Security Policy Requirement	Application Developer Tasks
Control whether the end user associated with a particular role may view or update **new set of data records for an existing business object** in the customized web page.	Create a new security policy. Although an existing Oracle Fusion business object will have an existing data security policy; you must not modify privileges granted to predefined duty roles (those defined by the Oracle Fusion security reference implementation) and you must instead grant only to custom duty roles that they define.	In the Oracle Fusion Data Security repository, add a custom duty role as the grantee of access privileges and create a named instance set for the new data records. Then, create the security policy by granting Oracle Fusion Data Security view or update privileges to the custom duty role for the data records. As a security guideline, do not modify a predefined data security policy by granting additional privileges to an predefined duty role.
Control whether the end user associated with a particular role may view or update **new *sensitive data*** exposed on a new attribute of an existing business object in the customized web page. *Sensitive data* is defined as any personally identifiable information (PII) that is considered "public within the enterprise" (also called "internally public"). Internally public PII data is secured from access external to the enterprise.	Create a new security policy. Sensitive PII data exposed by a new attribute that is added to an existing Oracle Fusion application business object will be secured by the business object's data security policies and will require a new data security policy to grant end users read or update access on a specific column of data.	Column-level OPSS authorization checking is not supported for ADF entity objects. Instead create a custom OPSS permission to control access to the column read or update operation, and then, in the Oracle Fusion Data Security repository, map the operation to a custom privilege and grant the privilege to the custom duty roles for the sensitive data records. Last, conditionally render the attribute by testing whether the end user has the custom privilege either 1.) by entering an ADF Security Expression Language (EL) utility method using an EL expression on the user interface component that displays the attribute or 2.) by entering a groovy expression on the ADF view object to which the user interface component is bound.
Control whether the end user associated with a particular role may view or update **new *confidential data*** exposed on an all new business object in the customized web page. *Confidential data* is defined as any personally identifiable information (PII) that is considered "private within the enterprise". Exposure of such information outside the enterprise could result in harm, such as loss of business, benefit to a competitor, legal liability, or damaged reputation. Confidential PII data is secured from access external to the enterprise and is secured additionally to prevent access within the enterprise by highly privileged end users (such as database administrators).	Create a new security policy. In Oracle Database, the Virtual Private Database (VPD) feature only supports securing a set of data records and therefore will require a new table in a custom schema that you create for Oracle Fusion Applications. The confidential PII data exposed by the new business object will be inaccessible by default (also called *protected*) and will require a new data security policy to grant end users read or update access on a specific set of data records.	Column-level policies are not supported by Virtual Private Database (VPD). Instead the database administrator must create a new table in your custom schema for Oracle Fusion Applications, create a view for that table, and then define a VPD policy to filter the PII data records by associating a PL/SQL function with that view. Then, in the Oracle Fusion Data Security repository, create an action with the same name as the database view and create the security policy by granting Oracle Fusion Data Security view or update privileges to the custom duty role for the confidential data records. Last, in the data model project, enable OPSS authorization checking on the appropriate operations of the ADF entity object (`read`, `update`, and `removeCurrentRow`) that maps to the new PII database table.

15.3.4 Scenarios Related to Extending and Securing Data Model Components

In Oracle Fusion Applications, when you want to extend the application to expose additional data, you create an ADF entity object and implement the operations that may be performed over a particular set of data records. The ADF entity object you create encapsulates the data as business object instances, corresponding to data records from a database table or view, such as an invoice or a purchase order. Typical operations are business functions like viewing, editing, or creating an instance of the business object.

Security concerned with controlling the operations that can be performed against specific data is called *data security*. Data security policies involve granting an end user, by means of the end user's membership in a role, the ability to perform operations on specific sets of data. For example, an accounts payable manager in the enterprise's western regional office may be expected to view and edit invoice data records but only for the customers in the western region. The Accounts Payable Manager role provisioned to the accounts payable manager authorizes access to the business functions required to view and edit invoices instances, and, in this case, the specific instances of the invoices business object striped for the western region.

Data security policies are implemented using Oracle Fusion Data Security, which is the technology that implements the security repository for data security policies. Oracle Fusion Data Security is implemented as a series of Oracle Fusion Applications database tables, sometimes referred to as *FND tables* (note that FND refers to resources in foundation tables) and includes tables like `FND_OBJECTS` that defines the protected database resource and `FND_GRANTS` that defines the access privileges for those database resources.

To protect the business object in the extended application, where it has been exposed as an ADF entity object, a database resource definition in the `FND_OBJECTS` table identifies the same table or view backing the ADF entity object. The *database resource* in Oracle Fusion Data Security is the data resource on which data security is enforced. After the business object is defined as an Oracle Fusion Data Security database resource, then a security policy must be created to grant access to the data records. The security policies for the database resource specify access privileges such as read, update, and delete privileges on specific sets of data records exposed by the business object.

Note: When an ADF entity object exposes a business object that does *not* require security, then no database resource for that business object needs to be defined in the Oracle Fusion Data Security repository. For complete details about Oracle Fusion Data Security, see the "Implementing Oracle Fusion Data Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

As an Oracle Fusion security guideline, a new data security policy must be created instead of modifying predefined data security policies of the Oracle Fusion security reference implementation. For example, a new data security policy is required to expose additional data records or operations for an existing business object. Additionally, a custom duty role must be created as the recipient of the new data security access privileges since granting privileges to a predefined duty role would alter the segregation of duties defined by the reference implementation.

Note: Developers are not entitled to modify the role hierarchy defined by the Oracle Fusion security reference implementation. Therefore, whenever you create a new duty role, you must consult the security administrator to assign the custom duty role to a job role or data role.

Additionally, the Oracle Fusion security reference implementation uses database-level security policies to protect most of the confidential *personally identifiable information* (PII), also called internally private data, that exists in the Oracle Fusion Applications schema. This type of security is implemented in Virtual Private Database (VPD) policies directly on the PII tables. In general, database administrators and other personnel with access to the database must not modify VPD policies implemented for Oracle Fusion Applications. However, when you create a business object that introduces confidential data and that data needs to be treated as internally private within the enterprise, then certain roles may be granted access to the confidential data for valid business reasons. For example, a human resources representative may require access to the employee's home addresses, while a dispatcher may require access to the home phone numbers of on-call staff.

Whether or not you will need to define a data security policy to grant access to data records depends on the type of customization, as summarized in Table 15–1. The scenarios for defining data security policies include the following.

When a new business object is introduced and it needs to be secured:

When you seek to secure additional data records in the extended application because a new ADF entity object is introduced, then an Oracle Fusion Data Security database resource must be defined to protect the data records and a new data security policy must be created to grant end users access to the data records exposed by the business object that the ADF entity object defines. The data records exposed by the business object will be unprotected (accessible to all end users) until a database resource identifying the business object is defined in the Oracle Fusion Data Security repository.

Note that the operations to be secured on the new business object will also require enabling OPSS authorization checking for those operations on the ADF entity object in the data model project, as described in Section 15.3.7, "What You Can Customize in the Data Model Project at Design Time."

When a new business object attribute is introduced and it maps to sensitive data:

When you modify an existing ADF entity object to include a new attribute that maps to data that not all end users need to view, then a new data security policy must be defined to grant end users access to the sensitive data. This is accomplished through a combination of a data security policy that grants a custom privilege and enforcement of the privilege in the application source.

Because Oracle Fusion Data Security does not support automatic enforcement of custom data security privileges, column-level security is not supported by default. You enforce the custom privilege in the application source by enabling OPSS authorization checking at the level of the user interface component or its databound ADF view object. Otherwise, without the custom data security privilege and custom privilege check, the data records (including the sensitive fields) exposed by the business object would be secured by the data security policy that already exists for the business object.

Important: Oracle Fusion Data Security alone will not prevent sensitive data from being accessed by highly privileged end users, such as database administrators. If the data needs to be treated as internally private (confidential data), then consider implementing additional security using Virtual Private Database (VPD) policies. However, do not implement column-level VPD policies to protect sensitive data exposed by attributes, as security for attributes is not supported by VPD in Oracle Fusion Applications.

When a new business object attribute is introduced and it maps to confidential data:

When you create an ADF entity object that introduces data that is to be treated as confidential to the enterprise, then define row-level VPD policies to control access to PII data by privileged users, including database administrators. Implementing VPD policies requires saving the confidential information in a new table in a custom schema for Oracle Fusion Applications.

In this case, the database administrator first creates the database table and the VPD policy to secure the PII data records. The VPD policy the database administrator creates associates a policy function (a PL/SQL function) with a particular database view or synonym. The policy function filters the rows for any query made against the PII data. Finally, you can proceed to create the actual data security policies by granting to an action that has been created with same name as the database view where the policy is defined.

For information about creating tables in a custom schema for Oracle Fusion Applications, see Section 11.8, "Customizing and Extending the Oracle Fusion Applications Schemas."

For information about creating VPD policies, see the "Using Oracle Virtual Private Database to Control Data Access" chapter in the *Oracle Database Security Guide*.

When new operations or new data records are introduced from an already secured business object:

When you introduce new operations or additional data records exposed by an existing ADF entity object into the extended application, you must not modify the predefined data security policies or data role templates that already exist for that business object. Instead create a new data security policy to grant end users access to the operations or data records that had previously remained protected.

Note that the operations to be secured on the business object may also require enabling OPSS authorization checking for those operations on the ADF entity object in the data model project, as described in Section 15.3.7, "What You Can Customize in the Data Model Project at Design Time."

When already exposed operations or data records need to be accessible by additional end users:

When you introduce functionality into the extended application that changes the access requirements of the operations and data records exposed by an existing business object, then those end users may be provisioned by existing job roles or data roles. Consult the security administrator to make the data accessible to additional end users through role provisioning. This type of customization does not require modifying the access privileges or the duty roles of an associated data security policy.

15.3.5 Scenarios Related to Extending and Securing User Interface Artifacts

When you want to extend an Oracle Fusion application user interface to support particular end user duties, you may either create a new ADF bounded task flow or customize an existing bounded task flow. The bounded task flow specifies the control flow that the end user is expected to follow when interacting with the web pages contained by the task flow. Similarly, top-level web pages (ones that are *not* contained by a bounded task flow) may be introduced or customized.

Security concerned with controlling access to a bounded task flow or top-level web page is called *function security*. Function security policies involve granting an end user, by means of the end user's membership in a role, the ability to access task flows and perform operations in the contained web pages. For example, the accounts payable manager must be granted access privileges to the task flow that provides the functions to manage the invoice data records. If the manager is authorized to access the task flow, then a data security policy governing the invoice records will specify the manager's right to access the actual data.

Function security is implemented at the most fundamental level as resource / action pairs which may be granted to secure specific application artifacts. Oracle ADF defines the actions needed to secure certain Oracle ADF application artifacts, including ADF bounded task flows and, in the case of top-level web pages, ADF page definitions files.

In the Oracle Fusion Applications environment, function security policies aggregate one or more resource /action pairs into an entitlement definition. The entitlement is the entity that is granted to a duty role. The function security policy for the Oracle ADF application artifact, confers the end user with function access rights, such as a such as view or manage, through a specific duty role.

The function security policies for all the resources of the Oracle Fusion application form the function security repository, which is implemented as an OPSS application policy store. The OPSS policy store in a test or production environment is an LDAP server running Oracle Internet Directory (OID). At runtime, OPSS performs authorization checks against the application policy store to determine the end user's access privileges.

Note: For more information about how Oracle Platform Security Services implements function security, see the "Understanding Security Concepts" part in the *Oracle Fusion Middleware Application Security Guide*.

The security administrator for the enterprise exports the LDAP-based application policy store for a particular Oracle Fusion application into an XML file-based policy store that allows you to add security policies using the tools provided by JDeveloper. As an Oracle Fusion security guideline, you must create a new function security policy rather than modify the predefined function security policies of the Oracle Fusion security reference implementation. Additionally, a custom duty role must be created as the recipient (also called the *grantee*) of the new function security access privileges since granting privileges to a predefined duty role would alter the segregation of duties defined by the reference implementation.

Note: Developers are not entitled to modify the role hierarchy defined by the Oracle Fusion security reference implementation. Therefore, whenever you create a new duty role, you must consult the security administrator to assign the custom duty role to a job role.

Whether or not you will need to create a function security policy to grant access to a task flow or top-level web page depends on the type of customization, as summarized in Table 15–1. The scenarios for defining function security policies include the following.

When a new task flow or top-level web page is introduced:

When you expose new functionality in the extended application through a new ADF bounded task flow or top-level web page that you create, then a new function security policy must be created to grant end users access to the application artifact.

The new ADF bounded task flow or top-level web page are the only use cases that require a new function security policy for the extended application.

When a new web page is introduced into an existing task flow:

When you modify an existing task flow to include new web pages, those web pages will be secured by the containing task flow's existing security policy. In this case, because all web pages contained by a bounded task flow are secured at the level of the task flow, there is no no need to add additional function security privileges specifically for the new page. You will, however, need to create a new data security policy to grant end users access to any new data records that were introduced by the customization.

When a web page is modified to display a new field of sensitive data:

When you modify a web page to display sensitive data for a single data record field (for example, by adding a column to a table component to display salary information), access to the field displayed by the component cannot be controlled by a function security policy (authorization checking is not implemented by OPSS at the level of ADF Faces user interface components in the web page). Instead, you enter ADF Security Expression Language (EL) utility methods on that part of the databound ADF Faces component responsible for rendering the field and test the end user's associated role.

Note that using EL expressions to conditionally render a portion of a user interface component does not control access to the actual data; truly sensitive data must be secured on the business object with a data security policy, as described in Section 15.3.4, "Scenarios Related to Extending and Securing Data Model Components."

When a web page is modified to display components that must not be viewable by all end users:

When you modify a web page to display components that not all end users need to view (for example, a button that deletes data records), access to the components cannot be controlled with a function security policy (authorization checking is not implemented by OPSS at the level of ADF Faces user interface components in the web page). Instead, you enter ADF Security Expression Language (EL) utility methods on the `rendered` property of the ADF component in order to hide or render the entire component based the end user's associated role.

Note that using EL expressions to conditionally render a user interface component does not control access to the actual data (if that component displays data); truly sensitive data must be secured on the business object with a data security policy, as described in Section 15.3.4, "Scenarios Related to Extending and Securing Data Model Components."

When existing task flows or top-level web pages must be accessible by additional end users:

When you introduce functionality into the extended application that changes the access requirements of an existing bounded task flow or top-level web page, then consult the security administrator to make the resource accessible to additional end users through role provisioning. This type of customization does not require changing the access privileges associated with the resource or the duties it defines.

15.3.6 What You Can Customize in the Data Security Policy Store at Design Time

Data security policies are stored in the Oracle Fusion Data Security repository and are defined and edited using Oracle Authorization Policy Manager, which you access through Oracle Fusion Functional Setup Manager, from the **Manage Data Security** task available in the Setup and Maintenance area of any Oracle Fusion Setup application.

> **Note:** After you select the **Manage Data Security** task in Oracle Fusion Functional Setup Manager, the environment redirects to the data security customization user interface provided by Oracle Authorization Policy Manager. In this document, although the data security customization tool is identified as Oracle Authorization Policy Manager, be aware that the tool must be accessed through Oracle Fusion Functional Setup Manager.

Data security policies control access to the database resources of an enterprise. Database resources in the Oracle Fusion security reference implementation include database tables and views and are predefined standard business objects that must not be changed. However, for cases where custom database resources must be secured business objects (defined by ADF entity objects in the data model project), you can be entitled to create custom duty roles, manage database resources, and create new data security policies using Oracle Authorization Policy Manager.

> **Important:** As an Oracle Fusion security guideline, the privileges granted by predefined data security policies assigned to duty roles of the reference implementation must not be changed by customization developers. Always create new data security policies to confer additional access privileges. Details about the Oracle Fusion security reference implementation can be found in the Oracle Fusion Applications security reference manuals.

The data security policy consists of privileges conditionally granted to a role in order to control access to instance sets of the business object. A privilege is a single action corresponding to an end user's intended operation on a single business object. A data security policy therefore is a grant of a set of privileges to a role on a business object for a given instance set. You can define the instance sets as a single row of data, multiple rows of a single table, or all rows of a single table.

The following security artifacts are recorded in the Oracle Fusion Data Security repository for a new data security policy:

- A database resource that references a primary key corresponding to the database table or view of the business object on which data security will be enforced.

After the database resource is defined in the data security repository, the Oracle Fusion Data Security implementation protects the data records and operations exposed by the business object by default, and a data security policy must be defined to grant end users access to the business object.

- One or more roles that will be assigned to the end users who can perform the granted actions.

  For more details about the roles used by Oracle Fusion Applications, see Section 15.2, "About Extending the Oracle Fusion Security Reference Implementation."

- A rule (also called a *named condition*) to define the available row instance sets in the form of a SQL predicate or simple filter (stored as XML) defined on the rows of the database resource.

  Instance sets may be a single row of data, multiple rows of a single table, or all rows of a single table. Only instance sets with multiple rows requires creating a named condition.

- One or more actions (such as view, edit, and delete) performed on database records that correspond to the operations supported by the business object (which may include custom operations).

At runtime, data security policies make data available to end users based on their provisioned roles according to the following means:

- Action grants that specify whether the end user has the right to perform the intended operation
- Condition evaluation for individual actions (and its corresponding operation) that specify the which data records from the database resource may be accessed

> **Note:** The application developer does not enforce data security policies when creating the policies. In the case of data security, you must enable OPSS authorization checking on each business object that needs data security. This enforcement is implemented in JDeveloper, as described in Section 15.3.7, "What You Can Customize in the Data Model Project at Design Time."

Related to data security is an Oracle Fusion security feature called the data role template. Oracle Fusion Applications supplies *data role templates* to anticipate typical Oracle Fusion security scenarios and to allow the enterprise to generate data security policies based on information that is specific to the enterprise, such as the names of business units on which to apply the data security policies. Typically, the implementation manager for Oracle Fusion Applications enters the template information and then runs the templates to generate data security policies and the supporting data roles.

When you create a new business object or expose a new set of data records in the extended application, you must confirm whether a data role template exists to generate data security policies for that business object. If a data role template exists, you can update the template to supply information pertaining to the business object, such as the data records to secure and the data dimensions to express stripes of data, such as territorial or geographic information used to partition enterprise data.

Using Oracle Authorization Policy Manager, you may perform the following data security-related customization tasks:

- Manage database resources.

  An existing database resource must not have its primary key altered, but you can define new named conditions and add new actions to map any new operations that you implement. If you create a new business object for a database table or view, you can create an all new database resource with named conditions (see below) and actions.

- Create named conditions to filter the rows of the business object. (Optional)

  The database resource conditions are specified as SQL queries that, when added to a security policy, filter the data and generate an instance set of available data records. Conditions specify the entitlements available to the end user for specific business objects. Conditions may be static or they may be parameterized to allow instance sets to be specified generically but granted specifically. Note a condition is only required when the data security policy does not secure either a single data record or all data records: both of these cases may be defined without named conditions when creating the security policy.

  Note that instance sets generated with parameters cannot be used for data security that is enforced declaratively, rather you need to write code to enforce OPSS authorization checking.

- Create data security policies consisting of privileges for a specific application role, named condition (optional), and business object.

  A privilege can map a standard action to a standard operation: read, update, and delete on a condition of a business object. The standard actions and the standard operations are named similarly.

  Alternatively, a privilege can map a custom action to a custom operation on a condition of a business object. The custom privilege, for example `ApprovePO`, is useful to secure a custom operation in the data model project or to secure any operation for row sets at the level of the individual ADF view object. The custom privilege also supports securing operations on columns through ADF Security Expression Language (EL) utility methods in the user interface project or Groovy scripting language expressions in the data model project.

  As an alternative to specifying a named condition, the data security policy can secure an instance set defined by a single data record or defined by all data records. Both of these cases may be selected when creating the data security policy.

- Generate data security policies by updating a data role template with data dimensions and data sets required to support the business object.

  A data role template generates data security policies for a business object based on supplied data dimensions to partition the data records into sets of data security policies. The template also maps instance sets for the data security policies it will generate to a particular data dimension. Instance sets are authored at the time the business object is registered as a database resource. Data dimensions and instance sets are specified as SQL clauses.

  Note that the SQL clauses cannot be modified after running the template.

For an overview of these tasks, see Section 15.4, "Defining Data Security Policies on Custom Business Objects." For detailed documentation, see the "Managing Oracle Fusion Applications Data Security Policies" and "Oracle Fusion Applications Data Role Templates" chapters in *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

15.3.7 What You Can Customize in the Data Model Project at Design Time

You create a data model project in JDeveloper to map custom business objects to ADF entity objects. At runtime, the ADF entity object creates a row set of data records exposed by the business object and simplifies modifying the data by handling create, read, update, and delete operations. In the data model project, you then define one or more ADF view objects on top of the ADF entity object to shape the data to the row set require by the tasks of the application, such as populating a form that displays a customer's sales invoice.

After you map the business object to an ADF entity object, enforcement of data security policies does not occur automatically on the data records of the exposed business object. The Oracle Fusion security approach protects the business object that has been registered as an Oracle Fusion Data Security database resource to ensure that end users do not have unintended access to sensitive data. In adherence of the security principle of protected by default, Oracle Fusion security separates defining policies and enforcing policies. Thus, by default, data security policies for a business object will remain ineffective until you enable OPSS authorization checking on the operations of the ADF business component. Enforcement of OPSS authorization checking can be specified either declaratively, at the level of ADF entity objects or ADF view objects, or programmatically, on any related code.

You can modify the data model project to opt into data security in two ways:

- At the level of the ADF entity object, to enable OPSS authorization checking on standard operations. Standard operations supported by ADF entity objects include, read, update, and delete current row. In this case, all ADF view objects based on the ADF entity object will have the same level of authorization checking enabled. The applicable data security policies will filter the data for each row set produced by these ADF view objects in exactly the same way.
- At the level of the ADF view object, to enable OPSS authorization checking on standard operations for a collection of rows. This provides a way to filter the data in the data model project based on an individual row set that the ADF view object defines. This level of authorization checking also supports defining a custom privilege (corresponding to the ADF view object read operation) in the data security policy store.

Using JDeveloper, you can perform the following security-related customization tasks in the data model project:

- Enforce row-level security for standard operations.

  Standard operations that you can secure are read, update, and remove current row. OPSS authorization checking is enabled directly on the ADF entity object to be secured. Although the ADF entity object maps to all instances of the business object, the data security policy defines conditions to filter the rows displayed to the end user.

- Enforce row-level security for custom operations.

  You may wish to enforce security for custom operations that are specific to the custom business object. Custom operations are not supported by ADF Business Components on the ADF entity object. When a data security policy defines a custom operation, you must enable it using a view criteria set on an ADF view object. The view criteria identifies the data security policy and business object.

- Enforce security for individual attributes of business objects.

  Column-level OPSS authorization checking is not supported on the attributes of ADF entity objects or ADF view objects. You must create a custom OPSS

permission for the column-level read or update operation and then map that to a custom privilege. Whether or not the user interface displays the column is specified by testing that custom privilege in the user interface using an EL expression on the secured attribute displayed by the user interface component.

For an overview of these tasks, see Section 15.5, "Enforcing Data Security in the Data Model Project." For detailed documentation, see the "Implementing Oracle Fusion Data Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

15.3.8 What You Can Customize in the User Interface Project at Design Time

Before you create function security policies, you will use JDeveloper to create a user interface project with the custom ADF bounded task flows or top-level web pages that you intend to secure.

To simplify the task of securing the functions of the extended application, ADF Security defines a containment hierarchy that lets you define a single security policy for the ADF bounded task flow and its contains web pages. In other words, the security policy defined at the level of the bounded task flow, secures the flow's entry point and then all pages within that flow are secured by the same policy. For example, a series of web pages may guide new end users through a registration process and the bounded task flow controls page navigation for the process.

Specifically, the Oracle ADF application artifacts that you may secure in the user interface project of the extended Oracle Fusion application are:

- ADF bounded task flow protects the entry point to the task flow, which in turn controls the end user's access to the pages contained by the flow

  The ADF unbounded task flow is not a securable application artifact and thus does not participate in OPSS authorization checks. When you need to secure the constituent pages of an unbounded task flow, you define policies for the page definition files associated with the pages instead.

- ADF page definition files associated with top-level web pages

  For example, a page may display a summary of products with data coordinated by the ADF bindings of the page's associated ADF page definition file.

For details about creating bounded task flows and databound top-level web pages, see the "Introduction to Building Fusion Web Applications with Oracle ADF" chapter in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Although you can create function security policies for the custom resources of the user interface project, enforcement of function security does not occur automatically. The Oracle Fusion security approach protects securable Oracle ADF application resources to ensure that end users do not have unintended access. In adherence of the security principle of protected by default, Oracle Fusion security separates defining policies and enforcing policies. Thus, by default, function security policies will remain ineffective until you enable OPSS authorization checking by running the ADF Security wizard in JDeveloper on the user interface project.

Important: After you run the ADF Security wizard, OPSS authorization checking is enforced on the bounded task flows and top-level pages. These application artifacts will be inaccessible when testing the application in JDeveloper. To enable access, you must create function security policies on the protected application artifacts, as described in Section 15.3.9, "What You Can Customize in the Application Security Policy Store at Design Time."

Using JDeveloper, you can perform the following security-related customization tasks in the user interface project:

- Enable OPSS authorization checking to protect Oracle ADF application artifacts.

  Oracle ADF application artifacts in the user interface project, including ADF bounded task flows and the top-level web pages (with a backing ADF page definition) will be protected when you configure ADF Security by running the ADF Security wizard with the **Authentication and Authorization** option selected. This ensures that end users do not have unintended access to sensitive work flows of the extended application.

- Conditionally display or hide components in the web page.

  ADF Security implements utility methods for use in EL expressions to access Oracle ADF application artifacts in the security context. For example, you can use the ADF Security utility methods to specify whether the end user is allowed to access create, edit, or delete buttons. Good security practice dictates that your application hides user interface components and capabilities for which the end user does not have access. For example, if the end user is not allowed access to a particular task flow, you can use the EL expression to evaluate the role membership of the end user to determine whether or not to render the navigation components that initiate the task flow.

For an overview of these tasks, see Section 15.6, "Defining Function Security Policies for the User Interface Project." For detailed documentation, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

15.3.9 What You Can Customize in the Application Security Policy Store at Design Time

You can use JDeveloper to add application security policies to a file-based policy store that the security administrator creates by exporting policies from the LDAP-based application security policy store. The file containing the exported policy store is the `jazn-data.xml` file.

As a security development guideline, only use JDeveloper tools to work on the exported file-based policy store, and do not edit the security definitions directly. JDeveloper supports iterative development of security so you can easily create, test, and edit security policies that you create for Oracle ADF application artifacts. In JDeveloper, you can also create end user identities for the purpose of running and testing the application in JDeveloper's Integrated WebLogic Server. You provision a few test end user identities with roles to simulate how the actual end users will access the secured application artifacts.

After testing in JDeveloper using Integrated WebLogic Server, you must consult with the security administrator to merge the LDAP-based application policy store in the staging environment with the security policies that you added to the exported XML file. Initially, the staging environment allows further testing using that server's identity store before deploying to the production environment. Thus, end user

identities created in JDeveloper are not migrated to standalone Oracle WebLogic Server and are used only in Integrated WebLogic Server to test the extended application.

> **Note:** For details about implementing and testing security using JDeveloper, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

The basic security artifact for function security is the JAAS (Java Authentication and Authorization Service) permission, where each permission is specific to a resource type and maps the resource with an allowed action. In general, the JAAS permission specifies the allowed operations that the end user may perform on a particular application artifact. However, from the standpoint of Oracle Fusion Applications, end users typically need to interact with multiple resources to complete the duties designated by their provisioned roles. To simplify the task of creating function security policies in the Oracle Fusion Applications environment, you work with OPSS entitlements to grant privileges for a variety of securable resources, including ADF task flows, web services, and SOA work flows to a role.

Function security policies that comprise entitlement grants with multiple application artifacts are called *entitlement-based policies*. Example 15–1 shows the Oracle Fusion Applications entitlement policy Maintain Purchase Orders which groups the OPSS permissions for ADF task flows, a web service, and a SOA work flow.

Example 15–1 Entitlement-Based Policy Groups OPSS Permissions as a Set that May Be Granted to a Role

```
Resource Type: ADF Taskflow
Resource: PO Summary
Action: view

Resource Type: ADF Taskflow
Resource: PO Details
Action: view

Resource Type: ADF Taskflow
Resource: Supplier Details
Action: view

Resource Type: Web Service
Resource: SpendingLimitCheckWS
Action: invoke

Resource Type: Workflow
Resource: POApproval
Action: submit
```

You use the security policy editor in JDeveloper to create the entitlement-based policy. JDeveloper modifies the source in the exported XML file. As Example 15–2 shows, entitlement-based policies in Oracle Fusion applications are defined in the `<jazn-policies>` element. The policy store section of the file contains a `<resource-type>` definition (that identifies the actions supported for resources of the selected type), a `<resource>` definition (to identify the resource instance that you selected from your application and mapped to a resource type), a `<permission-set>` definition (to define the resources and actions to be granted as an entitlement), and a `<grant>` definition with one or more entitlements (defined in the XML as a permission set) granted to the desired application roles (the grantee).

Example 15–2 Entitlement-Based Security Policy Definition in jazn-data.xml File

```
<?xml version="1.0" ?>
<jazn-data>
  <policy-store>
    <applications>
      <application>
        <name>MyApp</name>

        <app-roles>
        <app-role>
          <name>AppRole</name>
          <display-name>AppRole display name</display-name>
          <description>AppRole description</description>
          <guid>F5494E409CFB11DEBFEBC11296284F58</guid>
          <class>oracle.security.jps.service.policystore.ApplicationRole</class>
        </app-role>
      </app-roles>

      <role-categories>
        <role-category>
          <name>MyAppRoleCategory</name>
          <display-name>MyAppRoleCategory display name</display-name>
          <description>MyAppRoleCategory description</description>
        </role-category>
      </role-categories>

      <!-- resource-specific OPSS permission class definition -->
      <resource-types>
        <resource-type>
          <name>APredefinedResourceType</name>
          <display-name>APredefinedResourceType display name</display-name>
          <description>APredefinedResourceType description</description>
          <provider-name>APredefinedResourceType provider</provider-name>
          <matcher-class>oracle.security.jps.ResourcePermission</matcher-class>
          <actions-delimiter>,</actions-delimiter>
          <actions>write,read</actions>
        </resource-type>
      </resource-types>

      <resources>
        <resource>
          <name>MyResource</name>
          <display-name>MyResource display name</display-name>
          <description>MyResource description</description>
          <type-name-ref>APredefinedResourceType</type-name-ref>
        </resource>
      </resources>

      <!-- entitlement definition -->
      <permission-sets>
        <permission-set>
          <name>MyEntitlement</name>
          <display-name>MyEntitlement display name</display-name>
          <description>MyEntitlement description</description>
          <member-resources>
            <member-resource>
              <type-name-ref>APredefinedResourceType</type-name-ref>
              <resource-name>MyResource</resource-name>
              <actions>write</actions>
            </member-resource>
```

```
          </member-resources>
        </permission-set>
      </permission-sets>

      <!-- Oracle function security policies -->
      <jazn-policy>
        <!-- function security policy is a grantee and permission set -->
        <grant>
          <!-- application role is the recipient of the privileges -->
          <grantee>
            <principals>
              <principal>
                <class>
                   oracle.security.jps.service.policystore.ApplicationRole
                </class>
                <name>AppRole</name>
                <guid>F5494E409CFB11DEBFEBC11296284F58</guid>
              </principal>
            </principals>
          </grantee>
          <!-- entitlement granted to an application role -->
          <permission-set-refs>
            <permission-set-ref>
              <name>MyEntitlement</name>
            </permission-set-ref>
          </permission-set-refs>
        </grant>
      </jazn-policy>
    </application>
  </applications>
 </policy-store>
 <jazn-policy></jazn-policy>
</jazn-data>
```

While OPSS permissions granted for a single resource are not typically defined in the Oracle Fusion Applications environment, function security policies that use OPSS permissions for a single resource are called *resource-based policies*. Ultimately, a function security policy may have either one or more OPSS permissions, one or more OPSS permission sets (entitlements), but not both.

> **Note:** Granting access to web pages in Oracle Fusion Applications is enforced at the level of ADF Controller components called *bounded task flows*. Task flows in Oracle Fusion Applications are ADF Controller components that assemble the application's web pages (or regions within a web page) into a work flow that supports the tasks to be performed by application end users. Defining security policies on task flows instead of individual web pages is a security best practice that blocks end users from directly accessing the pages of a task flow. Web pages that are not contained in a task flow, are top-level pages and may have security policies defined individually.

Provisioning of end users with role membership is defined in the application's identity store and is a configuration task to be performed by the security administrator independent of security customization.

Using JDeveloper, you may perform the following function security customization tasks:

- Create an entitlement-based policy for all other application roles.

  An entitlement-based policy is a set of resource grants (set of OPSS permissions) that will be required by the end user to complete a task.

- Create a resource-based policy specifically for the built-in OPSS application role `authenticated-role`.

  A resource-based policy sets an OPSS permission on a single application resource and grants that permission to an application role. This type of function security is typically not used by securable resources in Oracle Fusion Applications. However, the resource-based policy must be used to make a custom resource accessible only to authenticated end users (ones who visit the site and logs in). For example, granting a view action permission to the built-in OPSS application role `authenticated-role` is the way to make an employee registration task flow accessible to all employees within the enterprise.

For an overview of these tasks, see Section 15.6, "Defining Function Security Policies for the User Interface Project." For detailed documentation, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

15.3.10 What You Cannot Do with Security Policies at Design Time

After you create the security policies, consult an IT security manager to migrate the policies to the staging environment.

The security manager is responsible for the following tasks.

- After testing is completed in JDeveloper, the security administrator must merge the file-based policy store with the application policy store in the staging environment.

  For information about how the security administrator merges the policies using Oracle Authorization Policy Manager, see the "Upgrading Oracle Fusion Applications Policies" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

- The security administrator must provision enterprise users by mapping enterprise roles (defined in the staging environment identity store) to the custom application roles.

  For information about how the security administrator provisions enterprise users using Oracle Authorization Policy Manager, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

- Before running the application in the staging environment, the security administrator must reconcile the application roles GUIDs of any data security policies that were created based on new custom application roles.

  When the file-based policy store is merged, the GUIDs of application roles are not preserved. For information about how the security administrator reconciles GUIDs in a staging environment, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

- Before running the application in the staging environment, the security administrator must modify the application to use the LDAP-based policy store provided by the testing environment.

  For more information, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

- After testing is completed in the staging environment, the security administrator can migrate the application policy store from the staging environment to the policy store in production.

  For information about how the security administrator migrates policies to a new environment, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

15.3.11 Before You Begin Customizing Security

Before you begin customizing security, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will need to do the following before you can begin customizing security:

1. Install JDeveloper and set up your development environment.

   For more information, see the "Setting Up Your Development Environment" chapter in the *Oracle Fusion Applications Developer's Guide*.

2. Create a customization application workspace.

   Before you can implement customizations using JDeveloper, you must create a workspace that imports the necessary parts of the application you want to customize. For more information, see Chapter 10, "Using JDeveloper for Customizations."

3. Start JDeveloper in the appropriate role.

   If you are implementing customizations on existing application artifacts, you must select the **Oracle Fusion Applications Administrator Customization** role when you start JDeveloper.

   If you are creating new custom application artifacts (such as, entity objects, view objects, and pages), you must select the **Oracle Fusion Applications Developer** role when you start JDeveloper.

4. Create the database resources in a custom schema for Oracle Fusion Applications.

   The database table exposes the data in your application. You are free to use any tool you wish to create database objects in your custom schema. For example, you may choose to work with the Database Navigator in JDeveloper to model database objects. For information about creating the table in a custom schema, see Section 11.8, "Customizing and Extending the Oracle Fusion Applications Schemas." For information about creating database objects, see the Designing Databases topics in the JDeveloper online help.

5. When securing confidential personally identifiable information (PII) create a new table in a custom schema for Oracle Fusion Applications, a view corresponding to the new table, and a VPD policy to associate a PL/SQL filter function with the view.

   The VPD policy filters the view to expose the data for which data security policies may be created. For information about creating the table in a custom schema, see Section 11.8, "Customizing and Extending the Oracle Fusion Applications Schemas." For information about creating VPD policies, see the "Using Oracle Virtual Private Database to Control Data Access" chapter in the *Oracle Database Security Guide*.

6. Obtain privileges to create or edit Oracle Fusion Data Security security policies.

   If you will be creating or editing Oracle Fusion Data Security security policies in Oracle Fusion Applications, you will need the correct privileges. When you have the correct privileges, Oracle Authorization Policy Manager allows you to access the security data security customization user interface. Contact your security administrator for details.

7. In Oracle Authorization Policy Manager, create custom application roles.

   Data security and function security permit granting access privileges to Oracle Fusion Applications duty roles (also called OPSS application roles). Although Oracle Fusion Applications ships with standard duty roles, as an Oracle Fusion security guidelines, you must create new duty roles rather than grant privileges to predefined duty roles.

   For information about creating application roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

8. In HCM core application, create job roles, as needed.

   Job roles (also called OPSS enterprise roles) provide access to application resources through the Oracle Fusion Applications role inheritance hierarchy which specifies the inherited duty roles. Although Oracle Fusion Applications ships with standard job roles, the security administrator can create a new job role when one does already exist that defines the new duties.

   The security administrator uses integrated Oracle Identity Management pages to create and manage job roles in Oracle Fusion Applications. For information about creating job roles, see the "Managing Roles" chapter in the *Oracle Fusion Middleware User's Guide for Oracle Identity Manager*.

9. Identify the business components in your application's data model project that you want to create or customize.

   You can create or customize ADF entity objects and ADF view objects using JDeveloper to expose business objects in your application and opt into data security policies. For information about creating these business components, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

10. Identify the application artifacts in your user interface project that you want to create or customize.

    The following application artifacts that you create or customize using JDeveloper may be secured: ADF bounded task flows and ADF page definition files for top-level web pages. For information about creating these artifacts, in Chapter 11, "Customizing and Extending ADF Application Artifacts."

11. In JDeveloper, run the ADF Security wizard on your application.

    When you run the ADF Security wizard it configures your application to enable authorization checking so that Oracle Platform Security Services (OPSS) running in Oracle WebLogic Server (and in JDeveloper's test environment, Integrated WebLogic Server) will utilize the security policies to authorize access to application resources by the end user. OPSS determines whether the end user (represented by the JAAS subject) has the privileges necessary to access the resources they intend.

    For information about running the wizard, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

15.4 Defining Data Security Policies on Custom Business Objects

In Oracle Authorization Policy Manager, the general process for defining a data security policy is as follows.

1. Register the custom business object as a database resource.
2. Define the instance set of data records that you want to associate with specific securable operations of the business component.

   An instance set in Oracle Fusion Data Security is a security artifact called a *condition*. The policy identifies conditions from the security repository to specify the row instance set available to the end user provisioned to the role with the right to perform the intended business component operation.

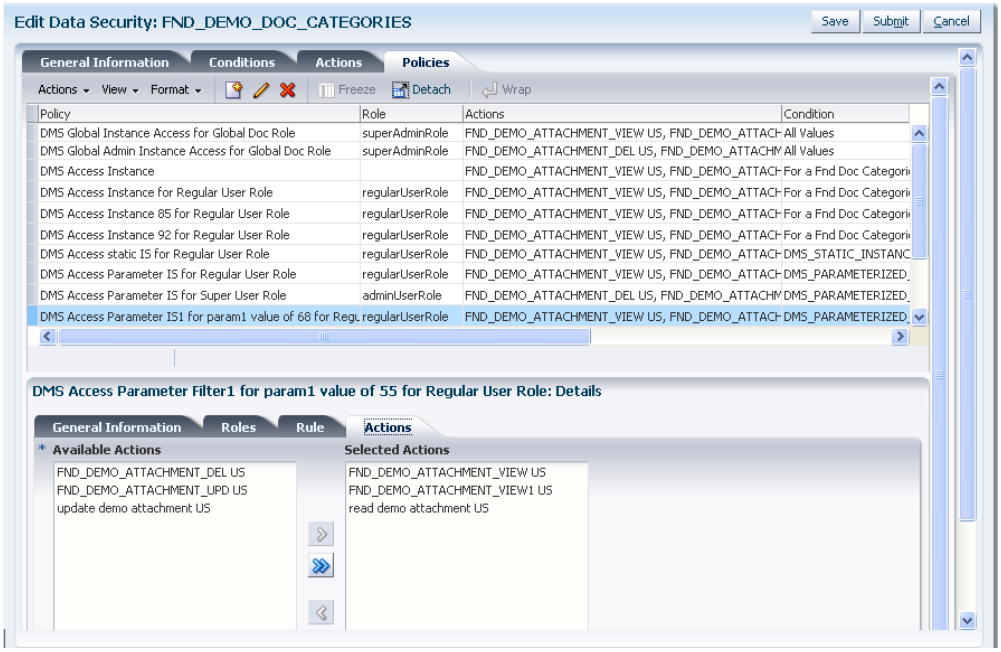
   In Oracle Authorization Policy Manager, a condition you create defines an instance set of multiple rows specified either by simple filters (XML defined) or complex SQL queries whose values can be parameterized. No condition definition is needed in the case of a single row instance or all the row instances of the database resource.
3. Define the list of actions that you want to be able to grant to the role.

   Action are database equivalent CRUD operations and correspond to the names of securable operations of the business object that the end user may invoke. The data security policy you define will associate one or more actions with an instance set.
4. If the custom business object is not supported by a data role template, define the data security policy:
   a. Enter a name and start date for the data security policy.
   b. Select one or more job roles or duty roles to which the policy grants access. The roles you select entitle all end users assigned to those roles with access to the data.

      In Oracle Authorization Policy Manager, duty role names that you enter are identified as OPSS internal roles called *application roles*. Similarly, job role names are identified as OPSS external roles called *enterprise roles*.
   c. Specify an instance set on the database resource for which the policy will control access. This may be a single row, all rows, or multiple rows (specified by a previously defined named condition).
   d. Specify one or more actions to secure on the database resource for the currently specified instance set.
   e. Repeat the steps to grant actions access to additional instance sets for the current data security policy and roles.

      Figure 15–1 illustrates the **Actions** tab in the Edit Data Security page after several actions have been selected. Available actions will be limited to the actions that had been defined for the database resource.

Figure 15–1 Creating a Data Security Policy - Selecting Actions



5. If the custom business object is supported by a data role template, then update the data role template with the following information:

   a. When the job role grantee of the data security policies generated by the template are not already defined by the existing data role template, add a new external role.

      The data role template specifies which base roles to combine with which dimension values for a set of data security policies.

   b. When the custom business object expresses a new stripe of data to apply to the generated data security policies, modify the SQL code that identifies the dimension of the template.

      Note that the SQL code cannot be modified after running the template.

   c. When the data role grantee of the data security policies generated by the template are not already defined by the existing data role template, configure a new data role name.

      The data role template constrains the data roles with access privileges for specific data records with particular actions. The data role provides provisioned end users with access to a dimensional subset of the data granted by a data security policy.

   d. Select the database resource that you registered for the custom business object.

   e. Optionally, select one or more data sets that you specified as named conditions when you created the database resource.

      Alternatively, the template can generate policies based on the primary key of the database resource.

   f. Specify one or more actions to secure on the database resource for the currently specified instance set.

Before you begin:

If you will be creating or editing Oracle Fusion Data Security security policies in Oracle Fusion Applications, you will need the correct privileges. When you have the correct privileges, Oracle Authorization Policy Manager allows you to access the security data security customization user interface. Contact your security administrator for details.

The Oracle Fusion security reference implementation defines the job role `IT Security Manager` with a duty role hierarchy that includes the `Application Data Security Administration Duty` duty role. This duty role is entitled to manage data security policies (the entitlement is `Manage Data Security Policy`) and provides the access necessary to perform the **Manage Data Security Policies** task in Oracle Authorization Policy Manager. Please contact your security administrator for details.

Additionally, collect the following information that you will use to define the data security policy in Oracle Authorization Policy Manager:

- The primary key of the database table or view that the custom business object represents.

  You specified the primary key of the database table or view when you registered the database resource.

- The names of the conditions for which you want the policy to control access.

  When you registered the database resource, you may have created named conditions to control access to instance sets comprised of multiple rows (Oracle Fusion Data Security does not require that you create a named condition when you want to grant access to instance sets comprised either of a single row or of all rows of the database resource).

- The names of the actions for which you want to associate with a particular named condition (or instance set) to control access.

  When you registered the database resource, you named actions to identify the securable operations of the custom business object. Action names must be identical to the names of the operations the business object supports. For example, the names of actions corresponding to the supported standard operations are **view**, **edit**, and **delete**. However, if your data model project defines custom operations, actions may have names corresponding to operations named, for example, as `view_US_ONLY`, `edit_US_ONLY`, or `delete_US_ONLY` .

- The names of the custom duty roles for which you want to grant access to the conditions and actions of the database resource associated with the custom business object.

  As an Oracle Fusion security guideline, predefined duty roles defined by the reference implementation must not be modified. You must use Oracle Authorization Policy Manager to create a new duty role rather than grant data security privileges to predefined duty roles. For information about creating roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Creating Conditions on a Business Object

A business object can define securable instance sets of data records called *conditions*. The data security policy you create may identify a specific data record, all data records of the object, or multiple data records. When you want to secure specific sets of records, then conditions must be created on the business object. From the

Administration menu in the global area of Oracle Fusion Applications, choose **Setup and Maintenance**, and then choose **Manage Data Security Policies**. After you register the business object as a database resource in the **General Information** tab, click the **Conditions** tab and click **New**. In the Create Database Resource Condition dialog, enter the SQL predicate consisting of a query on the table named by the database resource. For more information, see the "Managing Oracle Fusion Applications Data Security Policies" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Granting Access for a Privilege to a Specific Role and Object Condition

A business object can define conditions that query only the set of data records that are relevant to the members of a particular enterprise role or application role (also called job roles or duty roles, respectively). You can secure these sets of data records by making grants on conditions of the business object for a particular application role and privilege that you define. Condition-level security lets you secure any number of subsets of the business instances defined by the business object. As an alternative to standard privileges, you can define a custom privilege to create a security policy for operations that may be specific to a particular group of end users. Custom privileges also let you enforce security in the data model project at the level of the ADF view object and perform authorization checking to secure individual business object attributes. From the **Administration** menu in the global area of Oracle Fusion Applications, choose **Setup and Maintenance**, and then choose **Manage Data Security Policies**. After you register the business object as a database resource (using the **General Information**, **Conditions**, and **Actions** tabs), click the **Policies** tab and click **New**. You then use the policy workflow at the bottom of the Edit Data Security page (**Roles**, **Rule**, and **Action** tabs) to create the data security policy. For more information, see the "Managing Oracle Fusion Applications Data Security Policies" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Granting Access to a Specific Data Role and Dimension Values

A business object can be mapped to a set of dimension values and data role naming rules defined by data role templates. A data role for a defined set of data describes the job an end user does within that defined set of data. A data role inherits job or abstract roles and grants entitlement to access data within a specific dimension of data based on data security policies. The dimension expresses stripes of data, such as territorial or geographic information you use to partition enterprise data. You use data role templates to generate data roles and the template applies the values of the dimension and participant data security policies to the group of base roles. From the **Administration** menu in the global area of Oracle Fusion Applications, choose **Setup and Maintenance**, and **Manage Data Role Templates**. In the data role template workflow, you use the tabbed pages (**External Role**, **Dimension**, **Naming**, and **Policies** tabs) to create a data role template or revise an existing one. For more information, see the "Oracle Fusion Applications Data Role Templates" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

15.5 Enforcing Data Security in the Data Model Project

Data security policies secure data from business objects based on the grants made to roles. The business object participating in data security defines a database resource (a database table or view) that has been registered in the Oracle Fusion Applications `FND_OBJECTS` table. When you need to expose data records in the extended application you can use JDeveloper and Oracle ADF to create a data model project

with ADF entity objects based on secured database resources. However, it is not sufficient to register the business object in FND_OBJECTS and define data security policies, additionally, you must opt into those data security policies by enabling row-level OPSS authorization checking for specific operations on ADF entity objects in the data model project.

By default, after the database table or view backing the ADF entity object has been registered as a database resource in the FND_OBJECTS table, Oracle Fusion Data Security denies end users access to the business object data. Enabling OPSS authorization checking for the operations (such as view, edit, delete) by setting metadata on the ADF entity object of the data model project, ensures that only end users with sufficient privileges are authorized to perform the actions on the database resources corresponding to the ADF entity object.

JDeveloper saves the security metadata that you define on the data model project into an Oracle Metadata Services (MDS) repository.

Before you begin:

If the ADF entity object does not appear in the data model project, then you cannot opt into data security policies that may exist for the business object. You must use JDeveloper to create the ADF entity object based on a database table or database view that you intend to register in the Oracle Fusion Data Security schema. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

In order for OPSS to enforce security, the database table or view backing the ADF entity object must be registered as a business object with the FND_OBJECTS table provisioned by Oracle Fusion Data Security (the registered business object is also called a *database resource* of the Oracle Fusion Data Security schema). You will need to use Oracle Authorization Policy Manager to register the custom business object corresponding to the ADF entity object data source. For more information, see the "Managing Oracle Fusion Applications Data Security Policies" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Enabling security for custom operations in the data model project requires a custom privilege in the data security policy defined on the business object. You will need to create the custom privilege in the data security repository. For more information, see Section 15.4, "Defining Data Security Policies on Custom Business Objects."

Task: Enforcing Row Security for the Standard Operations of a Business Object

The ADF entity object in a data model project defines metadata that enables OPSS authorization checking against data security policies for view, update, or delete operations (also called standard operations) of the registered business object. You enable row-level security for standard operations by selecting the operation on the ADF entity object that maps to the business object upon which data security policies exist. Although the ADF entity object maps to all instances of the business object, the data security policy defines business object conditions to filter the records available to the end user. Filtering of the business object for standard operations supports only row-level security. In JDeveloper, you display the ADF entity object in the overview editor and, in the editor, click the **General** navigation tab and expand the Security section, and then you select the list of standard operations for which you want to enforce authorization checking against data security policies. For more information, see the "Implementing Oracle Fusion Data Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Enforcing Row Security for a Custom Operation of a Business Object

The ADF entity object in a data model project does not support OPSS authorization checking against data security policies for custom operations of the registered business object. You enable row-level security for custom operations by mapping a view criteria you create in the data model project to a custom privilege in the data security policy defined on the business object. The view criteria creates a row set filter by naming the custom privilege and business object. Filtering of the business object by view criteria works only with custom operations. In JDeveloper, you display the ADF view object in the overview editor and, in the editor, click the **Query** navigation tab and expand the View Criteria section, and then you click the **Add** button to create a view criteria to enforce authorization checking for a custom operation. For more information, see the "Implementing Oracle Fusion Data Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Enforcing Security for Attributes of a Business Object

The ADF entity object in a data model project does not support authorization checks against data security policies for columns of the registered business object. You enable security for attributes by creating a custom OPSS permission to control access to the column read or update operation, and then, in the Oracle Fusion Data Security repository, you map the operation to a custom privilege and grant the privilege to specify the roles that are authorized to view or update the data records. Last, in the user interface, you enter an ADF Security Expression Language (EL) utility method to test that custom privilege using an EL expression on the user interface component displaying the attribute. For more information, see the "Implementing Oracle Fusion Data Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

15.6 Defining Function Security Policies for the User Interface Project

You can use JDeveloper to define function security policies directly in an exported version of the Oracle Fusion application security repository. The security administrator exports the policies that exist in the LDAP-based application security policy store (residing in a test environment) into an XML file that can be loaded in JDeveloper and edited using the provided security policy editor.

After editing the XML file, you must consult the security administrator to merge the security policies into the test environment.

In JDeveloper, the general process for defining function security policies is as follows:

1. Consult an security administrator to export all predefined function security policies of the application that you are customizing into a `jazn-data.xml` file.

   For details about how the security manager exports the application policy store, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

2. Copy the exported `jazn-data.xml` file into your application workspace.

   This is the file that JDeveloper will update when you create function security policies. In order for JDeveloper to use the file, copy the file to your application workspace in the `<JDevAppHome>/src/META-INF` folder.

3. Create an entitlement to group one or more custom resources and their corresponding actions that together entitle end users to access the resource when needed to complete a specific duty.

In the Oracle Fusion Applications environment, the basic security artifact for entitlement-based security polices is the entitlement (an entitlement is defined by an *OPSS permission set*).

4. Grant the entitlement to a custom duty role that was added to the Oracle Fusion application policy store.

   The entitlement grant to the role specifies that the end user must be a member of the role to access the resources specified by the entitlement. You must use custom duty roles and you must not grant entitlements to predefined duty roles.

   In JDeveloper, duty role names that you select are identified as OPSS internal roles called *application roles*.

5. Enable ADF Security for the application by running the Configure ADF Security wizard.

   The wizard configures files that integrate ADF Security with OPSS on Integrated WebLogic Server.

   After you run the ADF Security wizard, any web page associated with a bounded ADF task flow will be protected. Therefore *before* you can run the application and test security, you must define the security policies that grant end users access.

Before you begin:

Consult the security administrator to obtain the file-based application policy store in the form of a `jazn-data.xml` file. The security administrator can run an Oracle WebLogic Script Language (WLSL) script to export the LDAP-based application policy store to the XML file. For more information about how the security manager exports the application policy store, see the "Securing Oracle Fusion Applications" chapter in the *Oracle Fusion Applications Administrator's Guide*.

If the custom bounded task flows or top-level web pages do not appear in the user interface project of the extended application, then you cannot define application security policies. You must use JDeveloper to create the securable Oracle ADF application artifacts. For more information, see Chapter 11, "Customizing and Extending ADF Application Artifacts."

As an Oracle Fusion security guideline, predefined duty roles defined by the reference implementation must not be modified. You must use Oracle Authorization Policy Manager to create a new duty role rather than grant function security privileges to predefined duty roles. For information about creating duty roles, see the "Managing Policies and Policy Objects" chapter in the *Oracle Fusion Middleware Oracle Authorization Policy Manager Administrator's Guide (Oracle Fusion Applications Edition)*.

Task: Defining Entitlement Grants for a Specific Application Role

An entitlement grant is a set of resource grants (set of OPSS permissions) that will be required by the end user to complete a task. Each permission in the entitlement grant names an OPSS permission class, a resource, and an action. Entitlements must be granted to custom application roles. In JDeveloper, when you want to define the privileges needed to perform a task, you choose **Application > Security > Entitlement Grants**, and then you name the entitlement, add member resources and the actions that you want to secure. Then you grant the entitlement to a custom application role. For more information, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

Task: Defining Resource Grants for the Authenticated User Role

A resource grant sets an OPSS permission on a single application resource and grants that permission to an application role. In JDeveloper, when you want to make the resource publicly accessible, you choose **Application > Security > Resource Grants**, and then you select the Oracle ADF artifact, the built-in OPSS role **authenticated-role** (or **anonymous-role**) as the grantee, and the action that you want to make public. For more information, see the "Implementing Function Security" chapter in the *Oracle Fusion Applications Developer's Guide*.

16

Translating Custom Text

This chapter describes how to localize the changes that you make to Oracle Fusion applications using Page Composer and CRM Application Composer. It also describes how to localize your navigator menu customizations and your flexfield and value set configurations.

This chapter includes the following sections:

16.1 About Translating Custom Text

If your Oracle Fusion Applications are running in different locales, you can localize your customizations such that end users see the custom text in the language of their locale. End users set their locale when they log in. Users can also set their locale through the **Personalization > Set Preferences** menu item in the Oracle Fusion Applications global area.

You use XML Localization Interchange File Format (XLIFF) documents stored in the Metadata Services (MDS) metadata repository to provide locale translations for your Page Composer, CRM Application Composer, and navigator menu changes. For flexfield and value set configurations, you provide locale translations using the appropriate maintenance tasks.

For information about XLIFF documents, see the "Manually Defining Resource Bundles and Locales" section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

16.2 Translating Resource Bundles from Metadata Services Metadata Repository

You can use the Oracle WebLogic Scripting Tool (WLST) `exportMetadata` command to obtain XLIFF documents and you can use the WLST `importMetadata` command to import XLIFF documents into the metadata repository. For information about the metadata repository and the `exportMetadata` and `importMetadata` commands, see the

Managing the Metadata Repository chapter in the *Oracle Fusion Middleware Administrator's Guide*.

> **Tip:** You can also use Oracle Enterprise Manager Fusion Applications Control to import and export the XLIFF documents from the metadata repository. For more information, see the "Transferring Metadata Using Fusion Middleware Control" section in the *Oracle Fusion Middleware Administrator's Guide*. The referenced procedure describes using Fusion Middleware Control, but also applies to Fusion Applications Control.

For specific information about localizing Page Composer and CRM Application Composer customizations, see Section 16.3, "Translating Page Composer and CRM Application Composer Customizations." For specific information about localizing navigator menu customizations, see Section 16.4, "Translating Navigator Menu Customizations."

Task: Define Translations for MDS Metadata Repository

To localize the custom text, complete the following steps.

1. Use the WLST `exportMetadata` command shown in Example 16–1 to export XLIFF documents from the metadata repository to a directory of your choice.

Example 16–1 WLST exportMetadata Command

```
exportMetadata(application='application', server='server',
toLocation='directory-path',
docs='xlf-classpath', applicationVersion='version')
```

   Set the `docs` attribute to the class path for the XLIFF file. For example, use `/oracle/apps/resourcebundles/xliffBundle/FusionAppsOverrideBundle.xlf` for the base file for Page Composer and CRM Application Composer custom text. Use `/oracle/apps/menu/CustResourceBundle.xlf` for the base file for navigator menu custom text. Use the following format for the names of locale documents:

   ```
   basename_language[_country].xlf
   ```

   Replace *language* with the ISO 639 lowercase language code, such as `fr` for France. When applicable, replace *country* with the ISO 3166 uppercase country code. Country codes are necessary when one language is used by more than one country. For example, use `CustResourceBundle_zh_CN.xlf` for custom translations for Chinese in the People's Republic of China.

   Because all Oracle Fusion Applications use the same MDS partition, you can use any Oracle Fusion application as an argument for the `application` attribute when you export an XIFF file for text customizations.

2. Synchronize the entries in the XLIFF documents and provide the translated text in the `<target>` tags, as shown in Example 16–2.

Example 16–2 Sample Translation

```
<trans-unit id="ACCOUNTING_DISTRIBUTION">
  <source>Accounting Distribution</source>
  <target>Ventilation comptable</target>
  <note>Accounting Distribution</note>
</trans-unit>
```

3. Use the WLST `importMetadata` command shown in Example 16–2 to import the modified documents into the metadata repository.

Example 16–3 WLST importMetadata Command

```
importMetadata(application='application', server='server',
fromLocation='directory-path',
docs='xlf-classpath', applicationVersion='version')
```

Because all Oracle Fusion Applications use the same MDS partition, you can use any Oracle Fusion application as an argument for the `application` attribute when you import an XIFF file for text customizations.

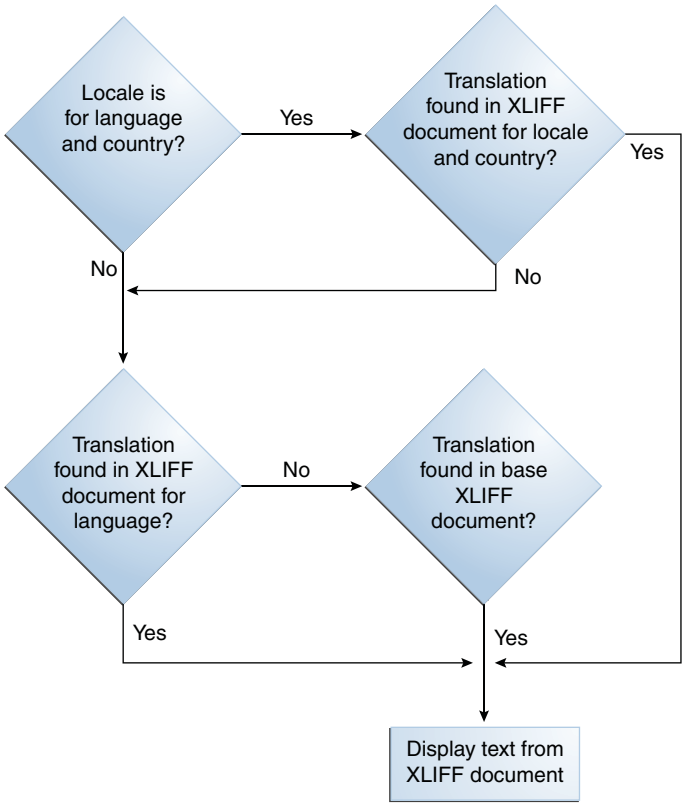
For more information about naming and editing XLIFF files, see the "Manually Defining Resource Bundles and Locales" section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

16.3 Translating Page Composer and CRM Application Composer Customizations

All Page Composer and CRM Application Composer customizations are stored in the customizations XLIFF document for the locale of the session in which you made the customizations. After you customize a page using Page Composer or CRM Application Composer, you might want to define translations for the custom text in the base customizations file as well as the customizations files for the other supported locales. For example, you might want to define French and Chinese translations of new prompts.

As shown in Figure 16–1, when an end user accesses the customized objects, the application loads the translated custom text for the locale's language and, if applicable, country. If the user's locale is for a language in a specific country and customized text is not available for that locale, the application loads the text for the locale's language. If no translated text is found, the application loads the text from the base customizations document.

Figure 16–1 Process for Retrieving Translated Text



Note that if no entries exist in the locale and base documents, the text that is displayed varies. For example, for a field label, the application displays the attribute name. In other cases, no text is displayed.

To define translations for custom text, follow the steps in Task: Define Translations for MDS Metadata Repository in Section 16.2, "Translating Resource Bundles from Metadata Services Metadata Repository." Export the base document `/oracle/apps/resourcebundles/xliffBundle/FusionAppsOverrideBundle.xlf` and the documents for all the locales for which you want to define translations. The locale XLIFF documents are named `/oracle/apps/resourcebundles/xliffBundle/FusionAppsOverrideBundle_`*language*`[_`*country*`].xlf`. Replace *language* with the ISO 639 lowercase language code, such as `fr` for France. When applicable, replace *country* with the ISO 3166 uppercase country code. Country codes are necessary when one language is used by more than one country. For example, use `/oracle/apps/resourcebundles/xliffBundle/FusionAppsOverrideBundle_zh_CN.xlf` for custom translations for Chinese in the People's Republic of China.

Note: The `/oracle/apps/resourcebundles/xliffBundle/FusionAppsOverrideBundle.xlf` base document is automatically generated the first time that a string is inserted or customized using Page Composer or CRM Application Composer. Ensure that the bundle exists by inserting or customizing at least one string.

Copy the new and changed entries from the document for the locale with which you made the customizations into the base document and into the other locale documents. Provide the translations and import the modified documents into the metadata repository.

16.4 Translating Navigator Menu Customizations

All navigator menu customizations are stored in the `/oracle/apps/menu/CustResourceBundle.xlf` base XLIFF document regardless of your locale setting when you customized the menu. After you customize the navigator menu, you might want to define translations for your changes in the locales that you support, including the locale for the session in which you entered the custom text. For example, you might want to define French and Chinese translations of new menu items.

The process for retrieving translated text is the same as for Page Composer and CRM Application Composer, as shown in Figure 16–1, with the exception that if no entries exist in the locale and base documents, no text is displayed.

To create locale translations for your navigator menu changes, follow the steps in Task: Define Translations for MDS Metadata Repository in Section 16.2, "Translating Resource Bundles from Metadata Services Metadata Repository." Export the base document `/oracle/apps/menu/CustResourceBundle.xlf` and export the documents for all the locales for which you want to define translations. The locale XLIFF documents are named `/oracle/apps/menu/CustResourceBundle_`*language*`[_`*country*`].xlf`. Replace *language* with the ISO 639 lowercase language code, such as `fr` for France. When applicable, replace *country* with the ISO 3166 uppercase country code. Country codes are necessary when one language is used by more than one country. For example, use `/oracle/apps/menu/CustResourceBundle_zh_CN.xlf` for custom translations for Chinese in the People's Republic of China.

Copy the new and changed entries from the base document into the locale documents and provide the translations. Then import the modified locale documents into the metadata repository.

16.5 Translating Flexfield and Value Set Configurations

When you first configure a flexfield or segment, the translatable text that you enter, such as prompts and descriptions, is stored as the text for all installed locales. To translate the text for a particular locale, log in with that locale or use the Personalization menu in the global area to set the locale. Then update the translatable text in the flexfield using the appropriate task, as described in Section 5.5.1, "Configuring Descriptive Flexfields" and Section 5.5.2, "Configuring Extensible Flexfields." Your modifications only change the translated values for the current session's locale.

After you complete the translations, deploy the flexfield as described in Section 5.7, "Deploying Flexfield Configurations."

You can define translations for Dependent and Independent value sets of type Character that have a subtype of Translated text. You define the translations by setting the current session to the locale for which you want to define the translation and using the Manage Value Sets task as described in Section 5.4, "Creating Custom Value Sets" to enter the translated values and descriptions for that locale.

For Table value sets for which the underlying table supports multiple languages and for which the value set's value column is based on a translated attribute of the underlying table, you can define translated values using the maintenance task for the

underlying table. For more information about enabling localization for Table value sets, see Section 5.4, "Creating Custom Value Sets." For information about multi-language support for tables, see the "Using Multi-Language Support Features" section in the *Oracle Fusion Applications Developer's Guide*.

17

Configuring End User Personalization

This chapter describes how you can make pages in your Oracle Fusion application personalizable by the end user. Note that mobile applications cannot be personalized by the end user.

This chapter contains the following sections:

17.1 About Configuring End User Personalization

Oracle Fusion applications allow end users to personalize certain pages using the Personalize menu. End users can set preferences, edit the current page, and reset the page to the default.

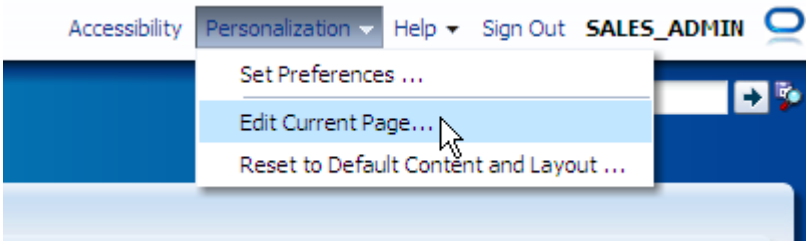
You can control what pages in an application can be personalized, including any new pages you may create.

> **Tip:** If you created a page using CRM Application Composer, then that page is personalizable by default.

> **Note:** For a list of pages that end users can personalize, refer to the product-specific documentation in Oracle Fusion Applications Help.

Figure 17–1 shows the Personalize menu available in all Oracle Fusion applications.

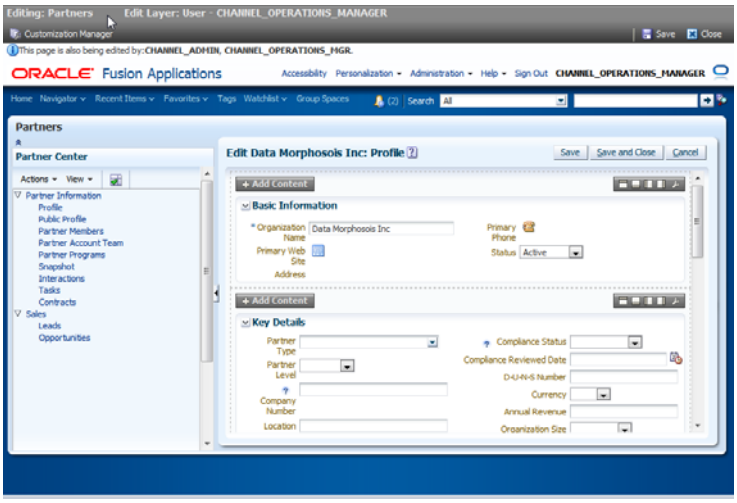
Figure 17–1 Personalization Menu in Oracle Fusion Applications



When end users choose the **Edit Current Page** menu item, Page Composer is launched. From here, they can change certain aspects of the page, such as moving or

deleting components. Figure 17–2 shows the Partner Profile application home page in Page Composer, ready for the end user to personalize.

Figure 17–2 Home Page Ready for Personalization



Along with using Page Composer to personalize pages, end users can change certain aspects of components, and then have those changes saved so that they remain each time the user logs into the application. For example, end users can change the width of columns in many of the tables in Oracle Fusion applications. However, by default, when they change the width, that new width size is only saved for the current session. You can configure that column so that when the user changes the width size, it will remain at that size whenever the user logs back into the application. For more information about configuring persistence, see "Chapter 35 Allowing User Customizations at Runtime" of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

17.1.1 Before You Begin Allowing Pages or Components to be Personalized

Before you configure pages to be personalizable, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will also need to do the following:

- Install Oracle JDeveloper and set up your development environment. For more information, see the "Setting Up Your Development Environment" chapter in the *Oracle Fusion Applications Developer's Guide*.
- Create a customization application workspace. For more information, see Chapter 10, "Using JDeveloper for Customizations."
- Launch JDeveloper in the Oracle Fusion Applications Administrator Customization role.
- Select a layer value. When customizing application artifacts in JDeveloper, you first need to select the layer and layer value to work in. You use the Customization Context window to make this selection. For more information about customization layers, see Section 1.2, "Understanding Customization Layers.""

17.2 Allowing Pages to be Personalized by End Users in Page Composer

You use JDeveloper to set certain attributes that allow a page to be personalized.

Task: Enable or Disable Personalization on Existing Standard Pages

Many pages in Oracle Fusion applications allow personalization by default. You can either disable it or enable it using the `isPersonalizableInComposer` property on a page. Set it to `true` to allow personalizations, set it to `false` to disallow personalizations. For instructions, see the "How to Enable End-User Personalizations for a Page" section of the *Oracle Fusion Applications Developer's Guide*.

Task: Enable Page Composer Personalization on Custom Pages

In order for end users to be able to use Page Composer to personalize custom pages, you'll need to enable your pages to work with Page Composer by doing the following:

- Set the `isPersonalizableInComposer` property to `true`.

  For instructions, see the "How to Enable End-User Personalizations for a Page" section of the *Oracle Fusion Applications Developer's Guide*.

  Note: If a page is currently available for personalizations, and you don't want it to be, change the value to `false`.

- Create a corresponding page definition file, if one does not exist.

  For instructions, see the "Ensuring Customizable Pages Have Page Definitions" section of the *Oracle Fusion Applications Developer's Guide*.

- Use Oracle WebCenter Portal components that define areas that can be customizable.

  For instructions, see the "Making a JSPX Document Editable at Runtime" section of the *Oracle Fusion Applications Developer's Guide*.

17.3 Configuring End User Personalization for Components

Certain attribute values that affect how an ADF Faces component displays can be persisted to the MDS repository. Application-wide component attribute persistence to the MDS repository is controlled by configuration in the `adf-config.xml` file. However, customizing this file is not allowed, as doing so is not upgrade-safe. Instead, you can override the application-wide persistence at the page level by setting the `persist` and `dontPersist` attributes for component instances.

For example, by default, table column attribute values are not persisted. But you can configure a column in a table so that when the user changes the width, reorders columns, or selects a column, those changes will still be in effect when the user logs back into the application, by adding those attributes to the value of the `persist` attribute on the column component. For more information about what attribute values can be persisted, see the "Introduction to Allowing User Customizations" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Note: You cannot change the settings in the `adf-config.xml` file, as these changes will be overwritten anytime you apply a patch or an upgrade. Therefore, you must change the values on the individual components on a page.

Task: Persist Attribute Values on JSPX Pages

You need to add the attributes you want to persist to the `persist` attribute on the component. For more information, see the "Controlling User Customizations in Individual JSF Pages" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. You can set this attribute using Page Composer. For more information about setting attributes on components, see Section 3.3, "Editing Component Properties in Page Composer."

18

Customizing Help

This chapter describes how you can customize or extend user assistance help in your Oracle Fusion application to match your runtime and design time customizations.

This chapter contains the following sections:

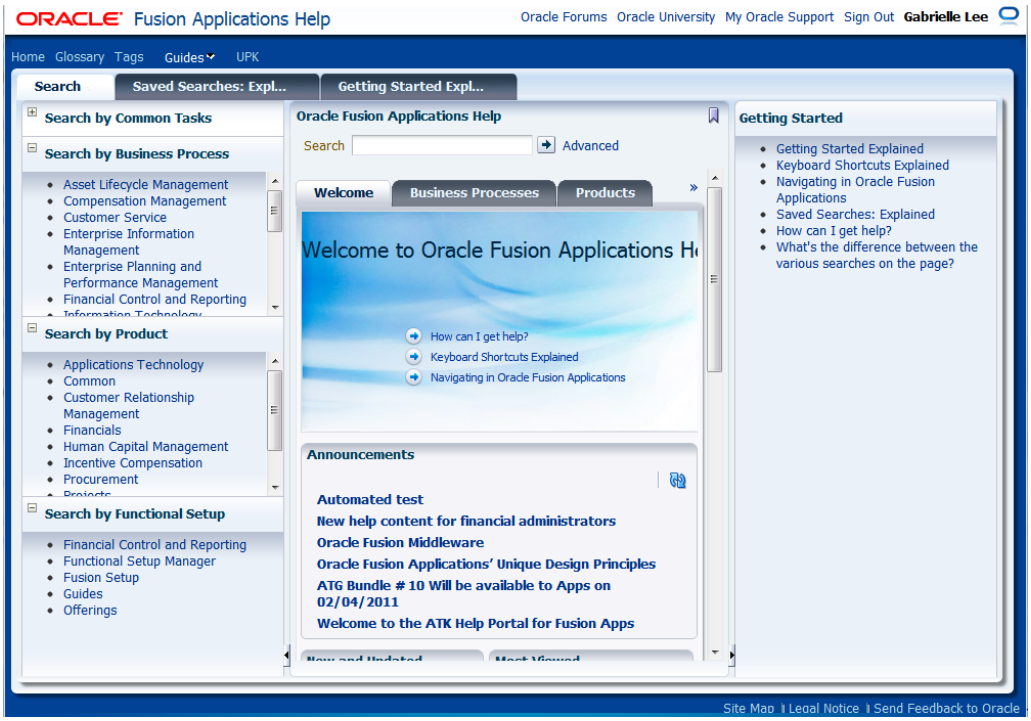
18.1 About Customizing Help

When you customize an Oracle Fusion application, you may find you also need to customize or extend the existing help to match your changes. Oracle Fusion applications provide two different types of help.

- Oracle Fusion Applications Help

  This type of help includes help topics, FAQs, examples, demonstrations and PDF guides, and is delivered with the Oracle Fusion Applications Help as shown in Figure 18–1.

Figure 18–1 Oracle Fusion Applications Help



- Embedded static page-level help

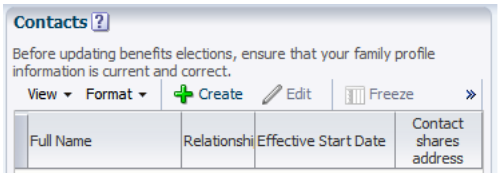
  This type of help is displayed directly on a page, using attributes of a component. The help text is included in the application.

  > **Tip:** Help text is stored in resource bundles, and so can be translated. For more information, see Chapter 16, "Translating Custom Text."

  Embedded help includes the following:

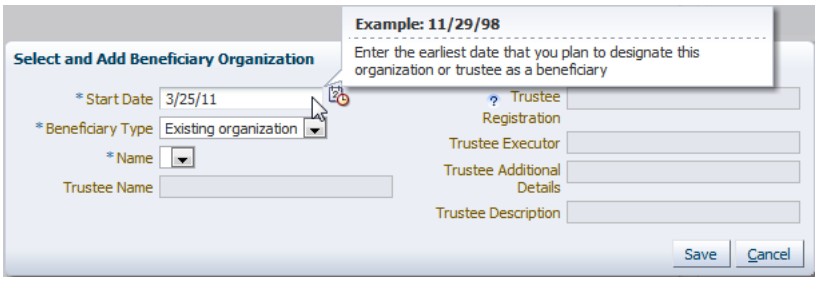
  – Static instruction text: displayed by panel components that normally contain forms or tables. This instruction guides the user in filling out the form or using the table, as shown in Figure 18–2.

Figure 18–2 Static Help Text in Oracle Fusion Applications



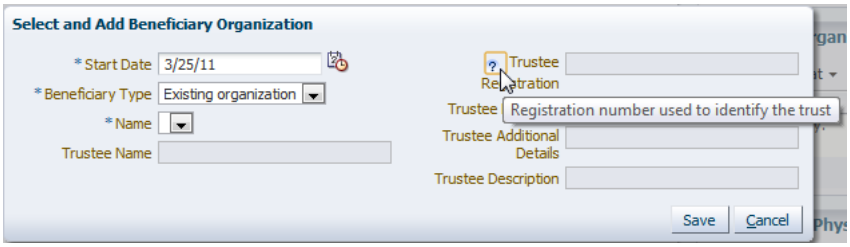
  – In-field help note: displayed by input components and guides the user in entering data into the component. Figure 18–3 shows an in-field note.

Figure 18–3 In-field Note in Oracle Fusion Applications



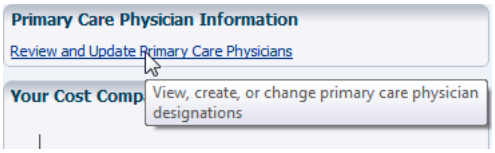
– Terminology definition: displayed by input components and defines terms used on the page, as shown in Figure 18–4.

Figure 18–4 Terminology Definition in Oracle Fusion Applications



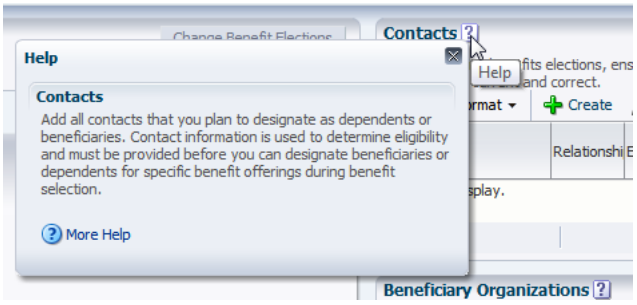
– Bubble help: displayed when the end user mouses over a button or link component, as shown in Figure 18–5.

Figure 18–5 Bubble Help in Oracle Fusion Applications



– Help window: displayed when a user clicks the help icon, as shown in Figure 18–6. This type of help is generally brief context-sensitive help, and can also provide links to help files in Oracle Fusion Applications Help.

Figure 18–6 Help Window in Oracle Fusion Applications



18.1.1 What You Can Do with Help

In Oracle Fusion Applications Help, you can change the content in existing help windows or you can create new help windows. Within a page of an application, you

can customize or create bubble help, static instructions, in-field notes, terminology definitions, and help windows. This help text is stored either as a value for an attribute, or in translatable resource bundles.

18.1.2 Before You Begin Customizing Help

Before you customize help, you should be familiar with the Oracle Fusion application architecture that enables customization, as described in Chapter 1, "Customizing and Extending Oracle Fusion Applications." You should also understand the typical workflows for working with customizations, as described in Chapter 2, "Understanding the Customization Development Lifecycle."

You will also need to do the following before you can begin customizing help:

- If you will be adding or customizing Oracle Fusion Applications Help, you will need the correct privileges. Please contact your security administrator for details.
- Install Oracle JDeveloper and set up your development environment. For more information, see the "Setting Up Your Development Environment" chapter in the *Oracle Fusion Applications Developer's Guide*.
- Create a customization application workspace. For more information, see Chapter 10, "Using JDeveloper for Customizations."
- Launch JDeveloper in the Oracle Fusion Applications Administrator Customization role.
- Select a layer value. When customizing application artifacts in JDeveloper, you first need to select the layer and layer value to work in. You use the Customization Context window to make this selection. For more information about customization layers, see Section 1.2, "Understanding Customization Layers."

18.2 Customizing or Extending Oracle Fusion Applications Help

You can customize existing help files in Oracle Fusion Applications Help, or you can extend Oracle Fusion Applications Help by adding custom topics.

Once created, custom help files are distinguished by an icon in search results, and they are displayed at the top of help listings when you navigate.

Task: Customize Oracle Fusion Applications Help Windows

When you have the correct privileges, help windows in Oracle Fusion Applications Help display a **Manage Custom Help** link, which allows you to change the content and specify in which help windows in the application your custom help will appear, and where it will appear in the help site navigators. For more information, see the "Define Help Configuration" section in the *Oracle Fusion Applications Common Implementation Guide*.

Task: Add Custom Help Files to Oracle Fusion Applications Help

You can add new custom help files to Oracle Fusion Applications Help. Custom help files will appear like standard help files and can be searched and included in help windows and navigators. For more information, see the "Define Help Configuration" section in the *Oracle Fusion Applications Common Implementation Guide*.

18.3 Customizing or Adding Bubble Embedded Help

For bubble help, you can use CRM Application Composer or Page Composer to customize or create the help text.

The following components use bubble help.

- `Butcon`
- `Button`
- `Link`
- `Tab`

Task: Customize or Add Bubble Help

The text displayed in bubble help is the value of the component's `shortDesc` attribute. Normally, the value resolves to a key in a resource bundle. If your are customizing a CRM application, you can use CRM Application Composer to customize the value of the attribute. For more information, see Section 4.2, "Editing Objects." For other applications, you use Page Composer to customize the attribute. For more information, see Section 3.3, "Editing Component Properties in Page Composer."

18.4 Customizing or Adding Static Instructions, In-field Notes, and Terminology Definitions

Oracle Fusion Applications embedded help (aside from bubble help) uses two types of ADF Faces help: *instruction* and *definition*. Instruction-type help displays static text, either in a specified area on a component (like static instruction help, shown in Figure 18–2), or in a note window, as in-field notes do, shown in Figure 18–3. Definition-type help displays a help icon, and is what terminology definition embedded help uses, as shown in Figure 18–4. When the user mouses over the help icon, the help text displays in a message box. UI components display the instruction and definition help text using the `helpTopicId` attribute. For more information about the ADF Faces help framework, see the "Displaying Help for Components" section of the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

It is important that for the type of help you want to add or customize, you understand which component actually displays the help, and which type of ADF Faces help is being used. Table 18–1 shows the different types of Oracle Fusion Applications embedded help, the corresponding ADF Faces help, and the components that display that type of help.

Table 18–1 Oracle Fusion Applications Help and Corresponding ADF Faces Help and UI Components

Oracle Fusion Applications Help Type	ADF Faces Help Type	Component
Static instruction	instruction	Page header
		Subheader
		Sub-subheader

Table 18–1 (Cont.) Oracle Fusion Applications Help and Corresponding ADF Faces Help and UI Components

Oracle Fusion Applications Help Type	ADF Faces Help Type	Component
In-field note	instruction	Multiselect check box group
		Single-select choice list
		Multiselect choice list
		Single-select list box
		Multiselect list box
		Text box
		Single-select radio groups
		Items in true/false radio groups
		Items in true/false check box groups
		Color picker
		Date/time picker
		Flexfield
		LOV
		Spin box
		Slider
		File upload
		Shuttle
		Rich Text Editor
Terminology definition	definition	Check box prompt
		Check box group prompt
		Single-select choice list
		Multiselect choice list
		Single-select list box
		Multiselect list box
		Text box
		Radio group prompt
		Color picker
		Date/time Picker
		Flexfield
		LOV
		Column headers
		Spin box
		Slider
		File upload
		Shuttle
		Rich Text Editor

You perform the following tasks in JDeveloper in the Oracle Fusion Applications Administrator Customization role.

> **Note:** You cannot directly customize the existing help text strings. If you want to change text that currently appears, you need to create a new text string and associate the component with that new text.

Task: Add Help Strings to Resource Bundle

Add custom help text strings to an existing custom resource bundle or create a new resource bundle to hold your customized help text (Oracle Fusion applications use XLIFF files for resource bundles). If you create a new resource file, you must register that file with the project. For information about creating and using resource bundles for an Oracle Fusion application, see Section 11.12, "Customizing or Adding Resource Bundles."

The help text must use the following syntax:

- `<trans-unit>`: Enter the topic ID. This must contain a unique prefix, the topic name, and the help type, either INSTRUCTION or DEFINITION.

  > **Note:** You prefix must unique. You must use this prefix for all your custom help strings.

  For example:

  ```
  MYCUSTHELP_NEWHELPTOPIC_DEFINITION
  ```

  In this example, MYCUSTHELPis the prefix used to access the XLIFF file. NEWHELPTOPIC is the topic name, and DEFINITION is the type of ADF Faces help.

  UI components access the help content based on the topic name. Therefore, if you use the same topic name for two different types of help (instruction and definition), then both types of help will be displayed by the UI component.

- `<source>`: Create as a direct child of the `<trans-unit>` element and enter the help text.
- `<target>`: Create as a direct child of the `<trans-unit>` element and leave it blank. This will hold translated text populated by translation tools.
- `<note>`: Create as a direct child of the `<trans-unit>` element and enter a description for the help text.

Example 18–1 shows an example of a resource file that contains two topics.

Example 18–1 XLIFF Resource Bundle

```
<?xml version="1.0" encoding="UTF-8" ?>
<xliff version="1.1" xmlns="urn:oasis:names:tc:xliff:document:1.1">
  <file source-language="en" original="this" datatype="xml">
    <body>
      <trans-unit id="MYCUSTHELP_NEWHELPTOPIC_DEFINITION">
        <source>Credit Card Definition</source>
        <target/>
        <note>This is the credit card definition text.</note>
      </trans-unit>
      <trans-unit id="MYCUSTHELP_NEWTOPIC2_INSTRUCTIONS">
        <source>Credit Card Instructions</source>
        <target/>
        <note>This is the credit card instruction text.</note>
```

```
      </trans-unit>
    </body>
  </file>
</xliff>
```

Task: Associate the Component with the Help Strings

In JDeveloper, select the component to display the help. Associate that component with the `<trans-unit>` element in the resource bundle, using the component's `helpTopicID` attribute. Ensure that the component supports the type of help (that is, definition or instruction) defined in the trans-unit id. For instructions, see the "How to Access Help Content from a UI Component" section of the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

19

Customizing the Oracle Fusion Applications Skin

This chapter describes how to use Oracle Application Development Framework (ADF) Skin Editor to change the look and feel of Oracle Fusion applications.

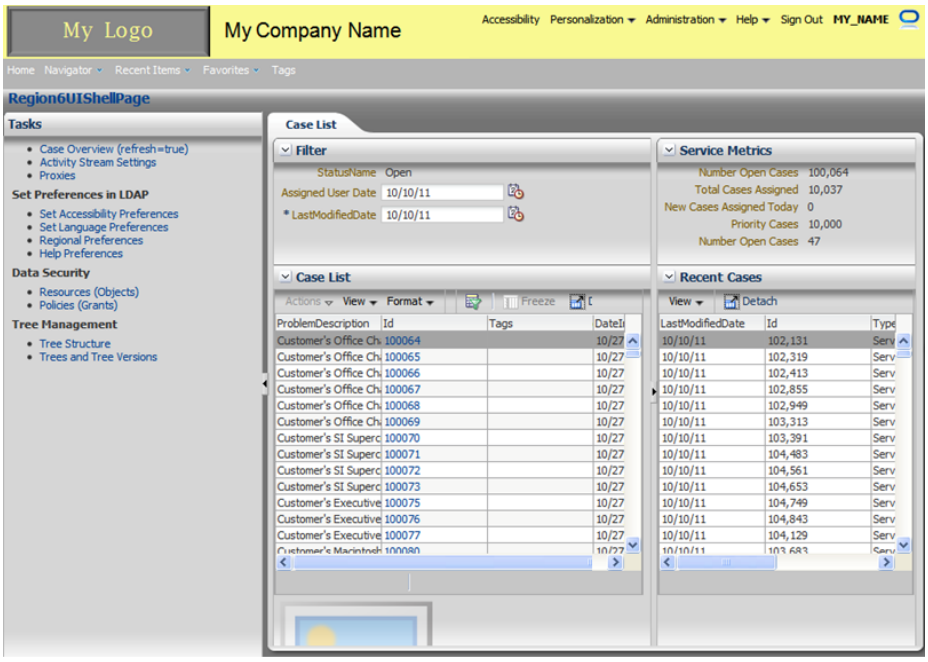
This chapter includes the following sections:

19.1 Introduction to Skinning Oracle Fusion Applications

If you want to make changes to the appearance of Oracle Fusion Applications pages, such as changing colors to make the pages adhere to your company's corporate brand, you can use ADF Skin Editor to create a custom skin based on the Oracle Fusion Applications Skin Extension (fusionFx-simple) and apply that skin to your Oracle Fusion applications. You can apply a custom skin to the whole site, to specific products, or specific end users. The changes that you make using custom skins are maintained through future patches and upgrades of Oracle Fusion Applications.

The fusionFx-simple skin extension is a special type of cascading style sheet (CSS) that enables you to customize the appearance of Oracle Fusion Middleware Extensions for Applications (Applications Core) components, ADF Faces components, and ADF Data Visualization components. Figure 19–1 shows an example of an application that has been skinned using the fusionFx-simple skin extension.

Figure 19–1 Example of a Skinned Application



19.1.1 Before You Begin Customizing the Oracle Fusion Applications Skin

Before you implement customizations in applications, you should be familiar with the ADF skinning frame work and how to use it to create a custom skin, as described in the "About Skinning a Web Application" chapter in the *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework*.

You should also be familiar with how to use the editor to work with Applications Core components as described in *Skinning Oracle Fusion Applications*, which is available at `http://www.oracle.com/technetwork/fusion-apps/tools/downloads/index.html`.

You will need to do the following before you can begin customizing the Oracle Fusion Applications skin.

1. Download ADF Skin Editor from Oracle Technology Network (OTN) at `http://www.oracle.com/technetwork/developer-tools/adf/downloads/index.html`
2. Install and start ADF Skin Editor as described in *Oracle Fusion Middleware Installation Guide for Oracle Application Development Framework Skin Editor*.
3. Select **Check for Updates** from the **Help** menu to install the most current release of the Oracle Fusion Applications Skin Extension (fusionFx-simple) as described in the "Working with Extensions" section in *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework*.

19.2 Creating a Custom Oracle Fusion Applications Skin

To create and modify a custom skin for your Oracle Fusion applications, use ADF Skin Editor to create a project, add a skin based on the fusionFx-simple skin extension, and modify the ADF Faces component, ADF Data Visualization component, and Application Cores component styles.

Task: Create a Custom Oracle Fusion Applications Skin

You create a skin by creating a new application in ADF Skin Editor and then creating a new ADF skin file in the project. Ensure that you set the project's target application release to the Oracle Fusion Applications release. When you create the ADF skin file, select the appropriate fusionFx-simple version from the **Extends** dropdown list. Make a note of the family name from the Create ADF Skin File dialog. You use this name when you apply the skin to your Oracle Fusion applications.

Task: Modify the Component Styles in the Custom Skin

As described in *Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework*, use ADF Skin Editor to change the look and feel of ADF Faces components, and ADF Data Visualization components. The fusionFx-simple skin extension additionally enables you to modify *Applications Core components*, which define how the Oracle Fusion Applications template and extensible components appear. You find the Applications Core components by expanding **Style Classes** in the Design view.

For information about modifying the look and feel of Applications Core components, see *Skinning Oracle Fusion Applications*, which is available at `http://www.oracle.com/technetwork/fusion-apps/tools/downloads/index.html`.

19.3 Applying a Custom Skin to Your Oracle Fusion Applications

When you are ready to apply your custom skin to your Oracle Fusion applications, you deploy the custom skin to an ADF Library JAR file. You then copy the custom skin JAR file plus supporting JAR files to the installation directories of the applicable Oracle Fusion applications, restart the applications, and set the profile option to use the custom skin.

Task: Deploy the Custom Skin to an ADF Library JAR File

You need to deploy the skin project to a JAR file that can be included in an application.

To create the JAR file:

1. Right-click the skin project, choose **Deploy**, and choose **New Deployment Profile** to display the Create Deployment Profile dialog.
2. Select **ADF Library JAR File** from the Profile Type dropdown list.
3. Set the **Deployment Profile Name** to a name that begins with `Xx_`. The `Xx_` prefix signifies to future patches and upgrades that this deployment is customer-provided and must not be touched.
4. Click **OK**.
5. Right-click the skin project, choose **Deploy**, and choose the profile name to display the Deploy *profile name* dialog.
6. Click **Finish**.
7. Right-click the skin project, choose **Deploy**, and choose *profile name* **to JAR file**.

Task: Add the Custom Skin JAR Files to Your Oracle Fusion Applications

You must make the custom skin JAR file and skin support JAR files available to Oracle Fusion Applications before you can apply the skin.

Copy the following JAR files to the `WEB-INF/lib` directory of every Oracle Fusion application:

- *skin-editor-installation-dir*`/jlib/adf-richclient-fusion-simple-`*version*`.jar`
- `XxApplCoreSkin.`*version*`.jar`. Download this JAR file from OTN at http://www.oracle.com/technetwork/fusion-apps/tools/downloads/index.html
- The ADF Library JAR file for your custom skin.

After you add the custom JAR files, you must stop and restart the Oracle Fusion applications as described in the "Starting and Stopping" section in the *Oracle Fusion Applications Administrator's Guide*.

Task: Apply the Custom Skin to Your Oracle Fusion Applications

You use the Manage Profile Option Values page in the Manage Profile Options task from the Setup and Maintenance work area to apply your custom skin to Oracle Fusion applications. You can set this value at the site, product, or user level. You typically set the option at the site level, however, if you want to test the skin, you can set it at the user level.

To use your custom skin, change the value of the FND_CSS_SKIN_FAMILY profile option to the skin family attribute that you set when you created your skin. If you do not know the skin family attribute, you can find it in the skin project's `trinidad-skins.xml` file.

Note: If you set the profile option at the site level, but you did not copy the necessary skin JAR files into the `WEB-INF/lib` directory of every Oracle Fusion application, the applications with the missing files will display with a simple skin that has a basic black-and-white look.

Part IV

Appendices

This part contains information about troubleshooting Oracle Fusion Applications extensions and customizations. It contains the following appendix:

- Appendix A, "Troubleshooting Customizations"

A

Troubleshooting Customizations

This appendix describes common problems that you might encounter when extending and customizing Oracle Fusion Applications and explains how to solve them. It contains the following topics:

- Introduction to Troubleshooting Customizations
- Getting Started with Troubleshooting and Logging Basics for Customizations
- Resolving Common Problems

In addition to this chapter, review the *Oracle Fusion Middleware Error Messages Reference* for information about the error messages you may encounter.

A.1 Introduction to Troubleshooting Customizations

Use the following guidelines and process for using the information in this chapter to help focus and minimize the time you spend resolving problems.

Guidelines

When using the information in this chapter, consider the following guidelines:

- After performing any of the solution procedures in this chapter, immediately retry the failed task that led you to this troubleshooting information. If the task still fails when you retry it, perform a different solution procedure in this chapter and then try the failed task again. Repeat this process until you resolve the problem.
- Make notes about the solution procedures you perform, symptoms you see, and data you collect while troubleshooting. If you cannot resolve the problem using the information in this chapter and you must log a service request, the notes you make will expedite the process of solving the problem.

Process

Follow the process outlined in Table A–1 when using the information in this chapter. If the information in a particular section does not resolve your problem, proceed to the next step in this process.

Table A–1 Process for Using the Information in this Chapter

Step	Section to Use	Purpose
1	Section A.2	Get started troubleshooting customizations. The procedures in this section quickly address a wide variety of problems.

Table A–1 (Cont.) Process for Using the Information in this Chapter

Step	Section to Use	Purpose
2	Section A.3	Perform problem-specific troubleshooting procedures for customizations. This section describes: ■ Common problems ■ Solution procedures corresponding to each of the possible problems
3	Section A.4	Use My Oracle Support to get additional troubleshooting information about Oracle Fusion Applications or Oracle Business Intelligence. My Oracle Support provides access to several useful troubleshooting resources, including Knowledge Base articles and Community Forums and Discussions.
4	Section A.4	Log a service request if the information in this chapter and My Oracle Support does not resolve your problem. You can log a service request using My Oracle Support at `https://support.oracle.com`.

A.2 Getting Started with Troubleshooting and Logging Basics for Customizations

This section provides the following general approaches for managing and diagnosing customization issues:

- Diagnosing Customization Issues Using the Manage Customizations Dialog
- Importing and Exporting Customizations
- Deleting Customizations
- Backing Up and Restoring Customizations
- Choosing the Right Customization Layer
- Determining the Full Path for a Customizations Document
- Determining Whether a Customization Layer is Active
- Logging Customizations that are Applied to a Page
- Determining Whether a Page has Customizations
- Using Sandboxes for Page Customizations
- Using Sandboxes for Flexfield Configurations
- Troubleshooting Flexfield Deployment
- Validating Flexfield Metadata

A.2.1 Diagnosing Customization Issues Using the Manage Customizations Dialog

The Manage Customizations dialog displays the customizations of the task flows in a page. You can access the Manage Customizations dialog from Page Composer and from the **Administration** menu in the global area of Oracle Fusion Applications. You can also use this dialog to delete page customizations, and to upload and download page customization files. For more information, see Section 2.3, "Viewing and Diagnosing Runtime Customizations."

A.2.2 Importing and Exporting Customizations

Customizations are stored in XML files. You can export the customizations in one of two ways.

- Use the Manage Customizations dialog to export user interface personalizations or page customizations. Choose the desired level and download the document.

  For more information, Section 2.4.1, "Downloading and Uploading Customization Files Using the Manage Customizations Dialog."

- Use an Oracle WebLogic Scripting Tool (WLST) command as shown in Example A–1.

Example A–1 WLST Command to Export Customization Document

```
exportMetadata (application='application name',
server='server name',
docs=
'/oracle/apps/hcm/dashboard/hrSpecialist/publicUi/page/mdssys/Site/SITE/VisaWorkPe
rmitExpirationRegion.jsff.xml',
toLocation='temp location');
```

For more information about using `exportMetadata`, see Section 2.4.2, "Downloading and Uploading Customization Files Using WLST Commands" and the "Application Metadata Management Commands" section of *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

For information about obtaining the string to use for the `docs` argument, see Section A.2.6, "Determining the Full Path for a Customizations Document."

> **Tip:** If you are not sure of the document name, append `'/*'` to the path in the `docs` argument to include all customization documents in the directory. Append `'/**'` to the path in the `docs` argument to also include the customization documents in the subdirectories. For example, use `'/oracle/apps/hcm/dashboard/hrSpecialist/publicUi/**'` to import or export all documents under the `publicUi` directory and its subdirectories.

You can also use Oracle Enterprise Manager Fusion Applications Control to export an application's customization files. For more information, see the "Transferring Metadata Using Fusion Middleware Control" section of the *Oracle Fusion Middleware Administrator's Guide*. The referenced procedure describes using Fusion Middleware Control, but also applies to Fusion Applications Control.

A.2.3 Deleting Customizations

You can use the Manage Customizations dialog to delete customizations. In the Name list, find the page that contains the customizations and click **Delete** for the customization document that you want to delete.

If the customizations were made by someone other than the logged-in user, and you have administrator privileges, you can display that person's customizations by selecting **Select User** from the **Layer Name** dropdown list.

For more information about the Manage Customizations dialog, see Section 2.3, "Viewing and Diagnosing Runtime Customizations."

You can also use Oracle WebLogic Scripting Tool (WLST) commands to delete customizations, as shown in Example 2–1.

Example A–2 WLST Command to Delete a Customization Document

```
deleteMetadata (application='application name',
server='server name',
docs=
'/oracle/apps/hcm/dashboard/hrSpecialist/publicUi/page/mdssys/Site/SITE/VisaWorkPe
rmitExpirationRegion.jsff.xml');
```

For more information about the `deleteMetadata` command, see the "Application Metadata Management Commands" section of *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

Alternatively, you can use Fusion Applications Control delete an application's metadata. For more information, see the "Managing the Metadata Repository" chapter in the *Oracle Fusion Middleware Administrator's Guide*.

A.2.4 Backing Up and Restoring Customizations

Before you make customizations, you can create a backup of a known good state by creating a label. If an issue occurs after the label, you can revert back to that label by promoting it to the tip. For more information, see the "Creating Metadata Labels" and "Promoting Metadata Labels" sections of the *Oracle Fusion Middleware Administrator's Guide*.

Another way to backup and restore customizations is by exporting and importing customization files, as described in Section A.2.2, "Importing and Exporting Customizations."

A.2.5 Choosing the Right Customization Layer

When you make customizations, be careful to choose the correct layer.

- Use the Site layer for customizations that affect all end users.
- Use the Global layer for Oracle ADF Business Component customizations.
- Use product specific layers appropriately as documented.

A.2.6 Determining the Full Path for a Customizations Document

The following string shows the structure of the full document path for a customization document:

`/package/mdssys/cust/layer-name/layer-value/document-name.suffix.xml`

For example, the full document path for the Visa Work Permit Expiration region is `/oracle/apps/hcm/dashboard/hrSpecialist/publicUI/page/mdssys/Site/SITE/VisaWorkPermitExpirationRegion.jsf.xml`.

You can obtain the full document path of a customized region on a page by completing the following steps.

1. Navigate to the page that contains the customized region and choose **Customize** *page_name* **Pages** from the **Administration** menu in the global area of Oracle Fusion Applications to open Page Composer.
2. If you have more than one layer available for customization, the Layer Picker dialog is displayed. In the **Edit** column, select the desired layer.

3. Choose **Source** from the **View** menu.
4. In the hierarchical list, drill down to and hover over the customized region to display the full document path of the JSF fragment that contains the customization, such as `/oracle/apps/hcm/dashboard/hrSpecialist/publicUI/page/mdssys/Site/SITE/VisaWorkPermitExpirationRegion.jsf.xml`. Make a note of this path.

For descriptive flexfield configurations, you can use the Register Descriptive Flexfields task to find the name of the flexfield's package.

A.2.7 Determining Whether a Customization Layer is Active

Customizations will not appear if the customization layer is not active in a product. To determine if a customization layer is active, open the `adf-config.xml` file for the application and look for the `<cust-config>` tag, as shown in Example A–3. The nested `<customization-class>` tags show the active layers.

Example A–3 Active Customization Layers

```
<adf-mds-config xmlns="http://xmlns.oracle.com/adf/mds/config">
<mds-config xmlns="http://xmlns.oracle.com/mds/config" version="11.1.1.000">
<cust-config>
  <match path="/">
    <customization-class name ="oracle.apps.fnd.applcore.customization.GlobalCC"/>
    <customization-class name ="oracle.apps.fnd.applcore.customization.SiteCC"/>
    <customization-class name ="oracle.apps.fnd.applcore.customization.UserCC"/>
  </match>
</cust-config>
</mds-config>
</adf-mds-config>
```

A.2.8 Logging Customizations that are Applied to a Page

To turn on runtime logging for customizations that are applied to a page, set the log level for the `oracle.mds.custmerge` module to FINEST. You can set the application's log level by choosing **Troubleshooting** from the **Help** menu. You might need to ask your administrator to give you permission to set the log level.

If you have administration privileges, you can also use Fusion Applications Control to set the log level.

A.2.9 Determining Whether a Page has Customizations

You can use logging, as described in Section A.2.8, "Logging Customizations that are Applied to a Page," or you can use the Manage Customizations dialog, as described in Section A.2.1, "Diagnosing Customization Issues Using the Manage Customizations Dialog," to determine whether customizations have been applied to a page.

If you suspect that a problem might have been caused by a customization on a page, such as a user interface component disappearing from a page, you can export the page's customizations as described in Section A.2.2, "Importing and Exporting Customizations" and examine the document file.

A.2.10 Using Sandboxes for Page Customizations

You should perform your page customizations in sandboxes and publish only after full testing. You can use any number of sandboxes for testing, but you should use just

one sandbox for publishing to prevent merge conflicts, as described in Section 2.2.1, "Sandboxes and Concurrent Usage." For information about analyzing merge conflicts, see Section A.3.2, "Sandbox Merge Conflict Detected but Not Resolved."

A.2.11 Using Sandboxes for Flexfield Configurations

When you deploy a flexfield, you have a choice of deploying to the full test environment or deploying to a sandbox. You should first deploy a flexfield to a sandbox so that you can test the configuration without affecting other end users. A flexfield that is successfully deployed to a sandbox will have a status of Deployed to Sandbox. After deploying to a sandbox, log out and log back in. The flexfield's sandbox will be active in your session for you to test. After testing, deploy to the full test environment so that the changes can be seen by other users. For more information, see Section 5.7, "Deploying Flexfield Configurations."

A.2.12 Troubleshooting Flexfield Deployment

After you deploy a flexfield using the Manage Descriptive Flexfields task or the Manage Extensible Flexfields task, look at the value in the Deployment Status column to ensure that its status is Deployed (or Deployed to Sandbox if you are testing in a sandbox). If not, review the message in the Deployment Error Message column. For more information, see Section 5.7, "Deploying Flexfield Configurations."

If you successfully deployed a flexfield, but the custom attributes do not appear on the user interface page, ensure that you logged out and logged back in after deploying the flexfield. The changes do not appear until you log back in.

In the case of an extensible flexfield, if a context does not appear in the user interface, verify that the context is associated with one of the category's pages.

To examine a flexfield's configuration, export the deployed artifacts using the `exportMetadata` WLST command as described in Section A.2.2, "Importing and Exporting Customizations."

A.2.13 Validating Flexfield Metadata

When you deploy a flexfield from the Manage Descriptive Flexfields task or the Manage Extensible Flexfields task, the metadata is validated and, if errors are found, the flexfield is not deployed. If you want to check whether a flexfield will pass validation, choose **Validate** from the **Actions** menu.

A.3 Resolving Common Problems

This section describes common problems and solutions. It contains the following topics:

- User Interface is not Displaying the Active Sandbox Customizations
- Sandbox Merge Conflict Detected but Not Resolved
- Dashboard Title Change Does Not Appear In Browser Title Bar, Navigator Link, Or Tab
- Cannot Launch Page after Personalizations
- Missing Navigator Menu Item
- Navigator Menu Item Does Not Work
- Customizations Context Table is Empty in Oracle JDeveloper

- Application Does Not Display Correctly After Applying a Customized Skin
- Nothing Changes After Clicking Cancel in Set Preferences Page

A.3.1 User Interface is not Displaying the Active Sandbox Customizations

Problem

The customizations that were made in the active sandbox are not appearing in the user interface.

Solution

Log out and log in again.

To ensure the sandbox customization cache is cleared, log out and log back in before you enter a sandbox and after you perform any of the following sandbox-related actions:

- Exit a sandbox
- Publish a sandbox
- Destroy a sandbox

A.3.2 Sandbox Merge Conflict Detected but Not Resolved

Problem

You created customizations in a sandbox using the guidelines suggested in Section 2.2, "Using the Sandbox Manager." When you published the sandbox, merge conflicts were detected but not resolved.

Conflicts between sandboxes can arise when there is more than one sandbox that is intended for publishing in use. If two sandboxes contain customization changes to the same artifact and both are being published, the sandbox that is published last is given an option (by the sandbox manager) to overwrite the changes for that artifact from the sandbox that was published first. If the user working in the second sandbox decides to force-publish the second sandbox, the changes published by the first sandbox are overwritten. These types of conflicts can also occur with shared metadata files such as resource bundles that store translatable strings.

For more information, see Section 2.2.1, "Sandboxes and Concurrent Usage."

Solution

Before forcing the publishing of the sandbox, complete the following steps to analyze the conflicts:

1. Use the Manage Customizations dialog to download the customization document from the mainline.
2. Make the sandbox active and use the Manage Customizations dialog to download the customization document from the sandbox.
3. Compare the two document files to analyze the conflicts.

A.3.3 Dashboard Title Change Does Not Appear In Browser Title Bar, Navigator Link, Or Tab

Problem

After using Page Composer to change a dashboard page title, the old name is still displayed in the browser title bar, navigator link, or tab.

Solution

In addition to changing the page title property in the Task List Properties tab in Page Composer, which affects the browser title, you must also make the following changes:

- Tasks lists menu entry, page heading, and tab title: Change the label in the Task List Task Properties tab as described in "Task: Customize a Page Title" in Section 3.2, "Editing a Page in Page Composer."
- Navigator menu entry: Use the Manage Menu Customizations task to change the label for the menu item, as described in Section 6, "Customizing the Navigator Menu."

A.3.4 Cannot Launch Page after Personalizations

Problem

After making personalizations to a page, an end-user cannot launch that page.

Solution

An administrator can use the Manage Customizations task to display and delete that user's personalizations for the page as described in Section A.2.3, "Deleting Customizations."

A.3.5 Missing Navigator Menu Item

Problem

An expected menu item is not appearing in the Navigator menu.

Solution

Verify whether the menu item has been hidden from view as described in Section 6.4, "Hiding and Showing Nodes."

A.3.6 Navigator Menu Item Does Not Work

Problem 1

A custom menu item was added and the browser is not displaying the page indicated by the URL.

Solution 1

Open the Manage Menu Customizations task and verify that the web application name is the same as the context root for the application, and that the view ID is the `id` attribute for the page's `<view>` tag in the product's `public_html/WEB-INF/adfc-config.xml` file. The URL should not contain the `.JSPX` suffix.

For more information, see Section 6.3, "Adding Items."

Problem 2

When you click an item in the navigator menu, you get a "webApp *value* not defined" error message.

Solution 2

Verify that the application is in the topology tables, as described in the "Viewing the Routing Topology of an Oracle Fusion Applications, Product Family, or Product" section in the *Oracle Fusion Applications Administrator's Guide*.

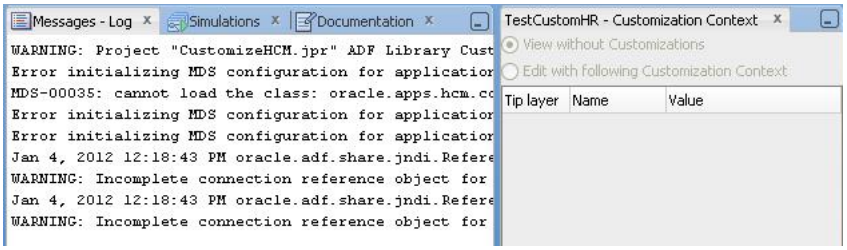
A.3.7 Customizations Context Table is Empty in Oracle JDeveloper

Problem

You are using JDeveloper in the Oracle Fusion Applications Administrator Customization role. The Customization Context table does not display the customization classes, as shown in Figure A–1, and the messages log displays an error message similar to the following text:

```
Error initializing MDS configuration for application
"file:/somepath/TestCustomHR.jws". Customizations disabled for this application.
MDS-00035: cannot load the class: oracle.apps.hcm.common.core.HcmCountryCC
```

Figure A–1 Empty Customization Context Table



Solution

Enable JDeveloper to see the customization classes that define the customization layers as described in Section 10.1.3, "Before You Begin Using JDeveloper to Customize."

A.3.8 Application Does Not Display Correctly After Applying a Customized Skin

Problem

After applying a customized skin that is based on the Oracle Fusion Applications Skin Extension (fusionFx-simple), the application does not show the expected customizations. For example, one or more of the following might occur:

- The background is not in the expected color.
- The user interface pages have a simple, minimal appearance instead of the expected skin.
- Expected images do not appear.

Solution

Verify that you used the correct target application version when you created the custom skin. Try repackaging and redeploying the jar and ensure that no problems occur during the packaging process.

Ensure that you copied the necessary JAR files to all the Oracle Fusion applications and that you spelled the name of the skin correctly in the profile option.

For more information, see Section 19.3, "Applying a Custom Skin to Your Oracle Fusion Applications."

A.3.9 Nothing Changes After Clicking Cancel in Set Preferences Page

Problem

You click the Cancel button in the Set Preferences page and nothing changes.

Solution

The cancel action resets the page to the same values that appeared when you accessed the page. If you have not made any changes, the page appears exactly the same. To navigate away from the page, choose **Recent Items** or **Navigator** from the global area of Oracle Fusion Applications or use the text box to search for a different page.

A.4 Using My Oracle Support for Additional Troubleshooting Information

You can use My Oracle Support (formerly MetaLink) to help resolve Oracle Fusion Middleware problems. My Oracle Support contains several useful troubleshooting resources, such as:

- Knowledge base articles
- Community forums and discussions
- Patches and upgrades
- Certification information

> **Note:** You can also use My Oracle Support to log a service request.

You can access My Oracle Support at `https://support.oracle.com`.