Oracle
Sales Cloud
Extending Sales

Release 13 (update 18B)
Contents

Preface

1 About This Guide
   Audience and Scope 1
   Related Guides 2

2 Configuring and Extending Oracle Sales Cloud
   Overview 3
   Run Time Changes 3
   Personalization 7
   Context Layers 8
   Changing Field Display Labels: Explained 12
   Business Process Models: Explained 14
   Enabling the Testing of Your Application Configurations 15

3 Application Changes Life Cycle
   Configuration Life Cycle: Explained 19
   Runtime Application Changes 20
   Sandbox Manager 22
   FAQs for Using Sandboxes 33
   Moving Application Changes 34
   Viewing and Deleting Configurations: Procedure 45
# 4 Adding Objects and Fields in Application Composer

<table>
<thead>
<tr>
<th>Using Application Composer: Overview</th>
<th>47</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extending Oracle Sales Cloud: How It Works</td>
<td>47</td>
</tr>
<tr>
<td>Viewing Application Composer Changes: Explained</td>
<td>50</td>
</tr>
<tr>
<td>Working with Objects</td>
<td>51</td>
</tr>
<tr>
<td>Managing Security for Custom Objects</td>
<td>64</td>
</tr>
<tr>
<td>Working with Fields</td>
<td>70</td>
</tr>
<tr>
<td>Actions and Links: Explained</td>
<td>108</td>
</tr>
<tr>
<td>Direct Page Links: Explained</td>
<td>113</td>
</tr>
<tr>
<td>Importing and Exporting Custom Objects: Explained</td>
<td>121</td>
</tr>
<tr>
<td>FAQs for Using Application Composer</td>
<td>122</td>
</tr>
</tbody>
</table>

# 5 Extending Simplified Pages

<table>
<thead>
<tr>
<th>Overview</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifying Oracle Sales Cloud Simplified Pages: Explained</td>
<td>127</td>
</tr>
<tr>
<td>Modifying Simplified Pages Using Page Composer: Procedure</td>
<td>129</td>
</tr>
<tr>
<td>Creating a Set of Simplified Pages for Custom Objects: Explained</td>
<td>131</td>
</tr>
<tr>
<td>Configuring a Search and Select Dialog Box: Explained</td>
<td>132</td>
</tr>
<tr>
<td>Working with Dynamic Page Layouts</td>
<td>134</td>
</tr>
<tr>
<td>Configuring the Summary Table on a Landing Page: Worked Example</td>
<td>153</td>
</tr>
<tr>
<td>Enabling or Disabling Drill Down Fields: Explained</td>
<td>154</td>
</tr>
<tr>
<td>Specifying Drill-Down Fields for Custom Dynamic Choice List Fields: Example</td>
<td>157</td>
</tr>
<tr>
<td>Creating and Adding Custom Links to Simplified Pages: Worked Example</td>
<td>158</td>
</tr>
<tr>
<td>Working with Subtabs</td>
<td>161</td>
</tr>
<tr>
<td>Working with Mashups</td>
<td>177</td>
</tr>
<tr>
<td>Modifying Work Area Lists</td>
<td>186</td>
</tr>
<tr>
<td>FAQ for Extending Simplified Pages</td>
<td>190</td>
</tr>
</tbody>
</table>

# 6 Extending Mobile Pages

<table>
<thead>
<tr>
<th>Overview</th>
<th>193</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle CX Cloud Mobile</td>
<td>193</td>
</tr>
<tr>
<td>Oracle Sales Cloud Mobile</td>
<td>199</td>
</tr>
</tbody>
</table>
## 7 Using Groovy Scripts

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>207</td>
</tr>
<tr>
<td>Groovy Scripting: Explained</td>
<td>207</td>
</tr>
<tr>
<td>Server Scripts: Explained</td>
<td>209</td>
</tr>
<tr>
<td>Global Functions: Explained</td>
<td>212</td>
</tr>
<tr>
<td>Accessing View Objects: Explained</td>
<td>213</td>
</tr>
<tr>
<td>Supported Classes and Methods: Explained</td>
<td>217</td>
</tr>
<tr>
<td>Using Groovy Scripting: Examples</td>
<td>234</td>
</tr>
<tr>
<td>Calling REST Web Services from Groovy Scripts</td>
<td>239</td>
</tr>
<tr>
<td>Calling SOAP Web Services from Groovy Scripts</td>
<td>256</td>
</tr>
<tr>
<td>Runtime Messages: Explained</td>
<td>282</td>
</tr>
<tr>
<td>Debugging Your Groovy Scripts: Explained</td>
<td>283</td>
</tr>
<tr>
<td>FAQ for Using Groovy Scripts</td>
<td>285</td>
</tr>
</tbody>
</table>

## 8 Creating Object Workflows

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>287</td>
</tr>
<tr>
<td>Object Workflows: Explained</td>
<td>287</td>
</tr>
<tr>
<td>Object Workflows: Examples</td>
<td>292</td>
</tr>
<tr>
<td>Object Workflows and Field Updates: How They Work Together</td>
<td>296</td>
</tr>
<tr>
<td>Object Workflows and E-Mail Notifications</td>
<td>297</td>
</tr>
<tr>
<td>Object Workflows and Tasks</td>
<td>302</td>
</tr>
<tr>
<td>Object Workflows and Outbound Messages</td>
<td>304</td>
</tr>
<tr>
<td>Object Workflows and Business Processes</td>
<td>309</td>
</tr>
<tr>
<td>Object Workflows and Groovy Scripts</td>
<td>344</td>
</tr>
<tr>
<td>Configuring Object Workflows: Worked Example</td>
<td>348</td>
</tr>
<tr>
<td>Deleting Unpublished Object Workflows</td>
<td>356</td>
</tr>
<tr>
<td>FAQs for Creating Object Workflows</td>
<td>357</td>
</tr>
</tbody>
</table>

## 9 Modifying Pages

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>359</td>
</tr>
<tr>
<td>Page Composer Overview</td>
<td>359</td>
</tr>
<tr>
<td>Modifying Page Content and Templates</td>
<td>367</td>
</tr>
<tr>
<td>Modifying User Interface Text</td>
<td>378</td>
</tr>
<tr>
<td>Modifying Themes</td>
<td>387</td>
</tr>
<tr>
<td>Modifying Infolets</td>
<td>394</td>
</tr>
</tbody>
</table>
## Using Page Composer in Oracle Sales Cloud 401
- Overview 401
- Configuring Oracle Sales Cloud UIs: Points to Consider 401
- Can I use Page Composer to configure all elements on a page? 402
- Configuring Oracle Sales Cloud Pages: Explained 404
- Modifying Simplified Pages: Procedure 412
- Changing Page Layout Using Page Composer: Procedure 414
- Configuring the Global Page Template: Explained 414
- What do I do if Page Composer quits while I am navigating in the UI? 416

## Understanding Analytics and Reports 417
- Providing Analytics for Sales Users: Overview 417
- Building Your Own Analytics: Explained 419
- Creating Your Own Subject Areas: Overview 421

## Configuration of Navigation 423
- Configuring Navigation and Home Page: Overview 423
- Navigator and Springboard Configuration 424
- Configuring Home Page Navigation: Procedure 430
- Using EL Expressions for Configuring Navigation: Examples 431

## Help Content Management 433
- Managing Help Content: Overview 433
- Who can create, edit, and manage help? 433
- Help Files Management 434
- Getting Started Pages Management 447
- Embedded Help Management 448

## Extending Activities and Sales Orders 449
- Overview 449
- Extending Simplified Pages for Activities: Explained 449
- Extending Simplified Pages for Sales Orders: Explained 451
### 15 Extending Account, Contact, and Household

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>453</td>
</tr>
<tr>
<td>Extending Pages for Accounts: Explained</td>
<td>453</td>
</tr>
<tr>
<td>Extending Pages for Contacts: Explained</td>
<td>455</td>
</tr>
<tr>
<td>Extending Pages for Households: Explained</td>
<td>458</td>
</tr>
<tr>
<td>Extending Simplified Pages for Relationships: Explained</td>
<td>460</td>
</tr>
<tr>
<td>Extending Pages for Assets: Explained</td>
<td>461</td>
</tr>
<tr>
<td>Extending Simplified Pages for Assets: Worked Example</td>
<td>462</td>
</tr>
<tr>
<td>Modifying the Overview Subtab: Explained</td>
<td>465</td>
</tr>
<tr>
<td>Enabling Display of Multiple Addresses: Explained</td>
<td>471</td>
</tr>
</tbody>
</table>

### 16 Extending Leads and Campaigns

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>475</td>
</tr>
<tr>
<td>Sales Lead and Sales Campaign</td>
<td>475</td>
</tr>
<tr>
<td>Copy Map</td>
<td>486</td>
</tr>
</tbody>
</table>
# 17 Extending Outlook

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>495</td>
</tr>
<tr>
<td>Configuring Outlook Pages Using Application Composer: Explained</td>
<td>495</td>
</tr>
<tr>
<td>Configuring Outlook Pages Using Application Composer: Procedure</td>
<td>496</td>
</tr>
<tr>
<td>Editing List Views: Procedure</td>
<td>497</td>
</tr>
<tr>
<td>Configuring Outlook Forms: Explained</td>
<td>497</td>
</tr>
<tr>
<td>Editing Form Fields: Procedure</td>
<td>498</td>
</tr>
<tr>
<td>Creating a Form Section: Procedure</td>
<td>498</td>
</tr>
<tr>
<td>Creating a Child Section: Procedure</td>
<td>499</td>
</tr>
<tr>
<td>Creating a Role-based Form Layout: Procedure</td>
<td>501</td>
</tr>
<tr>
<td>Synchronization Settings: Explained</td>
<td>501</td>
</tr>
<tr>
<td>Server Synchronization Filters: Explained</td>
<td>503</td>
</tr>
<tr>
<td>Setting up a Server-side Filter: Procedure</td>
<td>503</td>
</tr>
<tr>
<td>Creating a Role-based Synchronization Filter: Procedure</td>
<td>504</td>
</tr>
<tr>
<td>Editing Scripts: Explained</td>
<td>504</td>
</tr>
<tr>
<td>Creating a Script: Procedure</td>
<td>505</td>
</tr>
<tr>
<td>Configuring User-Defined Objects for Outlook Pages: Explained</td>
<td>506</td>
</tr>
<tr>
<td>Creating Outlook Pages for Parent User-Defined Objects: Procedure</td>
<td>507</td>
</tr>
<tr>
<td>Editing a Parent User-Defined Object List View: Procedure</td>
<td>508</td>
</tr>
<tr>
<td>Editing Parent User-Defined Object Form: Procedure</td>
<td>508</td>
</tr>
<tr>
<td>Adding User-Defined Child Object: Procedure</td>
<td>509</td>
</tr>
<tr>
<td>Editing Child User-Defined Object Form: Procedure</td>
<td>512</td>
</tr>
<tr>
<td>Adding Child Object to the Parent Object: Procedure</td>
<td>512</td>
</tr>
<tr>
<td>Defining Many-To-Many Relationships for Child Objects: Procedure</td>
<td>515</td>
</tr>
<tr>
<td>Selecting Outlook Setup Options: Explained</td>
<td>517</td>
</tr>
<tr>
<td>Synchronizing and Validating Changes: Explained</td>
<td>521</td>
</tr>
<tr>
<td>Managing Conflicts and Duplicates During Synchronization: Worked Example</td>
<td>521</td>
</tr>
</tbody>
</table>

# 18 Extending Partner Relationship Management

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>527</td>
</tr>
<tr>
<td>Extending Simplified Pages for Partners: Explained</td>
<td>527</td>
</tr>
</tbody>
</table>

# 19 Extending Territory Management

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring and Extending Simplified Pages for Territories: Explained</td>
<td>531</td>
</tr>
<tr>
<td>Configuring and Extending Oracle Sales Cloud Territory Management: Explained</td>
<td>532</td>
</tr>
</tbody>
</table>
### 20 Extending Sales and Opportunity Management 535
- Extending Sales and Opportunity Management Objects in Oracle Sales Cloud: Overview 535
- Extending Pages for Opportunities: Explained 535
- Extending Pages for Opportunities Using Application Composer: Worked Example 538
- Configuring Fields in Simplified UI Pages Using Page Composer: Worked Example 539
- Modifying Competitors Pages: Explained 542
- Modifying Your Object to Validate Assessments: Worked Example 544
- Hiding and Reordering Opportunity Fields Using Page Composer: Worked Example 545

### 21 Extending Sales Forecasting 551
- Overview 551
- Configuring and Extending Simplified Pages for Sales Forecasts: Explained 551
Preface

This preface introduces information sources that can help you use the application.

Using Oracle Applications

Using Applications Help

Use help icons to access help in the application. If you don’t see any help icons on your page, click your user image or name in the global header and select Show Help Icons. Not all pages have help icons. You can also access Oracle Applications Help.

Watch: This video tutorial shows you how to find help and use help features.

You can also read Using Applications Help.

Additional Resources

- **Community:** Use Oracle Cloud Customer Connect to get information from experts at Oracle, the partner community, and other users.
- **Guides and Videos:** Go to the Oracle Help Center to find guides and videos.
- **Training:** Take courses on Oracle Cloud from Oracle University.

Conventions

The following table explains the text conventions used in this guide.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates user interface elements, navigation paths, or values you enter or select.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates file, folder, and directory names, code examples, commands, and URLs.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than symbol separates elements in a navigation path.</td>
</tr>
</tbody>
</table>

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website.

Videos included in this guide are provided as a media alternative for text-based help topics also available in this guide.
Contacting Oracle

Access to Oracle Support
Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit My Oracle Support or visit Accessible Oracle Support if you are hearing impaired.

Comments and Suggestions
Please give us feedback about Oracle Applications Help and guides! You can send an e-mail to: oracle_fusion_applications_help_ww_grp@oracle.com.
1 About This Guide

Audience and Scope

This guide provides information on how sales administrators and implementors can make changes to Oracle Sales Cloud using the available configuration toolset. Administrators who modify applications should read this document to understand which tools to use, and how to use them, to make common application changes. In Oracle Sales Cloud, the configuration toolset includes Application Composer, Page Composer, BI Composer, and Business Process Composer.

Application Composer

Application Composer is a browser-based tool that business analysts and administrators, not just programmers, can use to modify Oracle Sales Cloud. Using this tool, you can make the types of data model changes which previously could only be made by application developers. For example, you can create a new object and related fields and then create new desktop pages to expose that object to users. Application Composer is a design time at runtime tool, which means that you can navigate to Application Composer directly from a Sales Cloud application, make your changes, and see most changes take immediate effect, without having to sign back into the application.

Note: Application Composer is supported for use only in English. Additionally, Application Composer is not supported for use with iPad devices.

Review each product-specific chapter in this guide to learn how to use Application Composer to make application changes for individual product areas.

Page Composer

Page Composer is a page editor that you can use to edit individual UI pages. In Oracle Sales Cloud, you can use Page Composer only for the following tasks:

- Creating work area lists (saved searches) for use by the whole sales organization or by individual roles.
- Reordering columns in search result tables.
- Configuring infolet pages.

The creation of work area lists and the reordering of columns is covered in the Extending Simplified Pages chapter of this book. You can learn how to configure infolets in the Managing Analytics in Work Areas chapter of the Creating and Administering Analytics for Sales guide.

BI Composer

The Oracle Business Intelligence (BI) Composer is a tool you can use to build reports. When building reports, you select a report subject area from within BI Composer. A report subject area is a set of entities, attributes, and measures that represent
information about the areas of an organization’s business. To build reports, use either the predefined report subject areas that are delivered for Oracle Sales Cloud, or create your own subject area using a wizard available in Application Composer.

To learn more about BI Composer, refer to the Understanding Analytics and Reports chapter in this guide.

Oracle Business Process Composer

When working with object workflows in Application Composer, you can define an object workflow that will trigger an approval flow if certain conditions are met. Approval flows are defined as business processes using Oracle Business Process Composer. Business Process Composer lets you orchestrate predefined components such as human-workflow tasks, services, and BPEL flows.

To learn more about Business Process Composer, refer to the Creating Object Workflows chapter in this guide.

Related Guides

You can refer to the following guides to learn more about making changes to Oracle Sales Cloud.

<table>
<thead>
<tr>
<th>Guide</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Sales Cloud Groovy Scripting Reference</td>
<td>Explains the basics of how to use the Groovy scripting language to enhance Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Oracle Applications Cloud Configuring and Extending Applications</td>
<td>Describes the tools and concepts for extending the applications.</td>
</tr>
</tbody>
</table>

Related Topics

- Oracle Help Center
2 Configuring and Extending Oracle Sales Cloud

Overview

Familiarize yourself with the types of changes that you can make using various configuration tools at either run time or design time, for all users or only some. Read this chapter to learn how you can modify your application using the different tools.

This chapter covers:

- How to personalize applications
- How to do runtime extensions
- How to make application changes using context layers
- What you can extend using which tool
- How to test role-specific application changes

In Oracle Sales Cloud, most user interfaces are implemented using Oracle Application Development Framework (Oracle ADF) and standard Java technologies. The foundation of the applications includes the service-oriented architecture (SOA) business processes. Business intelligence framework provides a number of reporting capabilities while the identity management works to control access at each level.

Oracle Sales Cloud is built using a common data model. Because of this commonality, when you make an application change in one area, that change is 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. Generally, the tools and processes you use to modify one application are the same tools and processes to modify all applications.

Run Time Changes

Runtime Configurations and Extensions: Examples

You can make configurations and extensions at run time using browser-based composers and other tools. All users or a subset of users can view and use these configurations and extensions. If your role has an administrative privilege, you can access most run time configuration tools to modify the user interface (UI), create and modify objects, and so on. Some configuration tools, such as Application Composer, are available only for specific product families.

Modifying the UI

To modify the UI, use:

- The User Interface Text page to edit text that appears on multiple pages. For example, you can change the term, "buyer" to "customer" if that is your preferred term, and the change affects all pages where the term is displayed.
• The Settings work area to change the:
  ◦ Look and feel of simplified UI
  ◦ Announcements on the home page

• Page Composer to configure simplified and desktop pages for other users. For example, you can:
  ◦ Add fields
  ◦ Add validation
  ◦ Change default content
  ◦ Rearrange regions
  ◦ Add external content
  ◦ Save queries

Tip: In Page Composer, you can make changes using the WYSIWYG view. However, in some cases, you can also use the Source view.

Configuring Navigation
Use the Structure work area to configure the Navigator and springboard. On the Navigator, select Configuration > Structure.

Adding User-Defined Attributes to Business Components Using Flexfields
Most business components, except those in Oracle Sales Cloud products, support using flexfields to add attributes to objects. Use flexfields to create your own attributes without programming. A flexfield captures data that is related to a specific purpose, such as information about job positions or inventory items. Each attribute is a segment of a flexfield, and corresponds to a reserved column in the application database.

Modifying Reports and Analytics
Predefined analyses, dashboards, and reports help in meeting business intelligence requirements. You can modify them to fit specific business needs, for example, change the layout.
For more information, see the Creating Analytics and Reports guides relevant to your products.

Managing Help
Use the Manage Help Content page to:
  • Add and edit help files in the application Help
  • Determine which help files to show in specific help windows
You can open the Manage Help Content page from any help window, or from the help site.

Note: You must have the appropriate job roles to add and edit help.

Related Topics
  • Help File Management: Overview
  • Flexfields: Overview
Tools for Configurations and Extensions: Critical Choices

You can configure and extend your application to suit your business needs. Choose an appropriate tool based on the types of configurations and extensions to make, such as:

- Page modifications
- Branding modifications
- Object modifications
- Security modifications
- Business intelligence modifications
- Help content management

⚠️ **Note:** The following tables present only the key tasks for application changes, not all tasks.

### Page Modifications

This table shows some types of modifications that you can make to pages, and the corresponding tools to use. You can modify only certain pages in Page Composer.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add, move, delete, show, or hide components on a page</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Change a page layout</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Create a site-level search for all users</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Change a page title</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Change a task list menu</td>
<td>Structure</td>
</tr>
<tr>
<td>Modify dialog box content</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Modify attributes for a flexfield on a page</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Change properties for user interface (UI) components on a standard page</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Configure the UI Shell template</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Update a text string wherever it appears across all pages</td>
<td>User Interface Text</td>
</tr>
<tr>
<td>Change the look and feel of application pages</td>
<td>Appearance page</td>
</tr>
</tbody>
</table>
Modification Task | Tool
---|---
Change the announcements on the home page | Announcements page

**Branding Modifications**
This table shows some types of modifications that you can make to use your own branding logo, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the UI Shell template</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Change the logo and application name in the UI</td>
<td>Appearance page</td>
</tr>
<tr>
<td>Change report layouts</td>
<td>Layout editor in the BI application or external applications such as Microsoft Word</td>
</tr>
</tbody>
</table>

**Object Modifications**
This table shows some types of modifications that you can make to objects, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add an attribute to a business object using flexfields (not Oracle Sales Cloud)</td>
<td>Setup and Maintenance work area</td>
</tr>
<tr>
<td>Add a business object page to the Navigator menu</td>
<td>Setup and Maintenance work area</td>
</tr>
</tbody>
</table>

**Security Modifications**
This table shows a security modification that you can make to objects, and the corresponding tool to use.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add data security to a custom object</td>
<td>Setup and Maintenance work area</td>
</tr>
</tbody>
</table>

**Business Intelligence Modifications**
This table shows some types of modifications that you can make to business intelligence (BI) analytics and reports, and the corresponding tools to use. For more information, see the Creating and Editing Analytics and Reports guides relevant to your products.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create report layout</td>
<td>Layout editor in the BI application or external applications such as Microsoft Word</td>
</tr>
</tbody>
</table>
Modification Task | Tool
---|---
Change report layouts | Layout editor in the BI application or external applications such as Microsoft Word
Create a report | The BI application
Modify analyses | Reports and Analytics pane or the BI application
Configure dashboards | The BI application

**Help Content Management**

This table shows some types of changes that you can make to help, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Modification Task</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify text that is displayed when the user hovers over a button, link, icon button, or tab title</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Manage help files and determine the help links to show on help windows</td>
<td>Oracle Fusion Applications Help</td>
</tr>
<tr>
<td>Simultaneously replace multiple occurrences of a word or phrase that appear in the embedded help</td>
<td>User Interface Text</td>
</tr>
</tbody>
</table>

Oracle Sales Cloud has additional tools available for application changes. For example, Application Composer and Business Process Composer.

**Related Topics**

- Flexfields: Overview
- Help File Management: Overview

**Personalization**

**Personalization: Explained**

Personalization refers to the changes that every user of the application can make to certain artifacts in the user interface (UI) at run time.

✏️ Note: Personalization changes remain for a user each time the user signs in to the application.
Personalization includes:

- Changes based on how you use the UI, such as changing the width of a column in a table
- Changes that you select to save, such as search parameters
- Changes you make to the springboard and infolets

Context Layers

Context Layers: Explained

Use the built-in context layers in your application to make changes that affect only certain instances or users of an application. Before making application changes, select the layer in which you want to make changes. Most of the configuration tools provide a dialog box for selecting the layer for your changes. If the dialog box to select a layer doesn’t appear before you make application changes, then by default your changes are made to the Site layer.

Available Layers

The context layers available to an application depend on its application family. However, all applications have the Site layer and User layer. Application changes made in the Site layer affect all users. Personalizations such as setting column size for a table or editing infolet titles are in the user layer.

Layer Hierarchy

The layers are applied in a hierarchy, and the highest layer in that hierarchy in the current context is considered the tip layer. An object may be modified more than once, but in different layers. At runtime, changes in layers closer to the tip-layer take precedence. For example, say you make application changes in the Site layer. You use Page Composer to add a region on a page. Then you use a layer for a specific job role to hide the region. In such a case, the job role specific changes take precedence for a user of that job role at runtime.

Storage of Configurations and Layer Information

Configurations aren’t saved to the base standard artifact. Instead, they’re saved in Extensible Markup Language (XML) files for each layer. These files are stored in an Oracle Metadata Services repository. The XML file acts like a list of instructions that determines how the artifact looks or acts in the application, based on the context layer. The configuration engine in Oracle Metadata Services manages this process.

When you apply an application patch or upgrade, it updates the base artifacts, but it doesn’t touch the configurations stored in XML files. The base artifact is replaced. Hence, when you run the application after the patch or upgrade, the XML files are layered on top of the new version. You don’t have to redo your application changes.

Working With Context Layers: Examples

The following scenarios illustrate working with context layers to ensure that the correct configurations or personalizations are available at run time to appropriate users. For example, job role is a layer. When you modify an artifact, you can choose to make that application change available only to users with a specific role, for example, a sales representative.
Configuring Pages for Users with Specific Roles

You want to remove the Quick Create panel from the Sales home page, and configure this page only for users with the Sales Representative role.

Following are the prerequisites:

- Activate a sandbox.
- When you modify a page for a specific job role, that role must be assigned to you for you to test the application changes in the sandbox. Your security administrator can either assign the role to you directly, or make the role self requestable for you to add it from the resource directory.
- Select the layer in which you want to make your application changes. In this case, select the role layer with the value, Sales Representative. While modifying pages, when you remove the panel from the page, an XML file is generated. This file contains instructions to remove the panel, but only for the role layer, and only when the value is Sales Representative.

**Note:** The original page file remains untouched.

The configuration engine in Oracle Metadata Services then stores the XML file in the Oracle Metadata Services repository. When someone signs in and requests an artifact, the configuration engine in the metadata service checks the repository for XML files matching the artifact and the given context. On matching, the configuration engine layers the instructions on top of the base artifact.

In this example, whenever someone:

- With the role of Sales Representative (the context) requests the Sales home page (the artifact), before the page is rendered, the configuration engine in Oracle Metadata Services:
  - Pulls the corresponding XML file from the repository
  - Layers it on top of the standard Sales home page
  - Removes the panel
- Without the role of Sales Representative signs in, the configuration engine doesn’t layer the XML file on top of the standard Sales home page. So, the Quick Create panel is displayed on the page.
This figure shows how the XML file with configurations is applied to the base document and is displayed only for a sales representative.

**Personalization**

Users can personalize their home pages and infolets. For example, you can:

- Personalize your home page content that you use to navigate in the application. Based on your requirements, you can show or hide the groups and page entries that appear on the home page.
- Edit infolet titles and views, move them around, and show or hide specific infolets on an infolet page.

While you personalize a page, the configuration engine in Oracle Metadata Services creates an XML file specific to a user (in this case, you), for the User layer. For example, say User 1 (with the role of Sales Representative) personalizes the Sales home page. An XML file, noting the changes that the user made, is stored in the repository. When User 1 signs in, the configuration engine in the metadata services:

- Pulls the XML file with the sales representative configurations from the repository, and layers the file on top of the standard Sales home page
- Pulls the XML file with the User 1 personalizations, thus enabling the user to see the personalization changes along with the Sales Representative changes.
When other sales representatives sign in, they don’t see the User 1 personalization changes, as shown in this figure.

Related Topics

- Autoprosioning: Explained
- How do I provision roles to users?
- How can I personalize the page entries on my home page?
- Personalizing Infolets: Procedure

Selecting Context Layers to Include: Examples

While making application changes, when you use the dialog box to select the context layer, you can also include lower layers to view the application changes from those layers.

The following scenarios explain what happens based on your selected layers. For these examples, the available layers are Site, Country, and Job Role.
What You See While Making Application Changes

Suppose you choose to:

- Edit the Job Role layer and select Sales Representative as the value for that layer
- Include the Country layer and select France as the value

Note: The Site layer is automatically included because it applies to everyone.

While modifying pages in Page Composer, you see changes that apply to sales representatives in the France country, based on:

- What was defined for each layer
- Which is the highest layer with application changes for a specific artifact

What Your Application Changes Apply to

No matter what you see while making application changes, your changes apply only to the selected layer for your changes, that is, Job Role. For example, say a field is hidden in the Site layer, but displayed in the Country layer for France. No changes exist for the field in the Job Role layer for sales representative. Since Country is higher than Site, you see the field displayed while modifying pages in Page Composer. However, if you choose to hide the field as part of your changes, then that change applies to the Job Role layer for sales representatives. Users with other job roles in the France country may still see the field. However, Job Role is higher than Country. So, no sales representatives in any country can see the field, unless a layer higher than Job Role applies to any of these users and has the field displayed.

Changing Field Display Labels: Explained

You can change the display labels (strings) that appear in Oracle Sales Cloud using Application Composer and User Interface Text Tool. These tools enable the administration of strings. Your tool selection depends on the scope of the changes that you want to make. For large-scale changes or changes to labels in locales other than English, use the User Interface Text Tool. For changes on a single page or to all labels for a custom object in the English locale, use Application Composer. Read this topic to learn more about best practices for changing display labels.

Display Labels

A single field label, such as Products, can occur throughout Oracle Sales Cloud on a variety of pages. However, just because the same field label appears multiple times doesn't mean that it's the same field across all pages. In fact, it's probably not. Note that there is a difference between a field label instance and a field label value. The value is what is seen on a page. The instance is the actual field location where the label value is retrieved from. Typically, multiple display label instances may exist that have the same value. This is especially true for common labels such as Account and Customer.

For example, you want to change Products to Our Products, but the field label Products appears throughout Oracle Sales Cloud 500 times. You don't have to change the Products display label 500 times. However, you will probably need to change it more than once. Again, this is due to the underlying architecture of Oracle Sales Cloud where multiple display label instances can (and do) exist with the same value.

Display Labels and Multiple Languages

Oracle Sales Cloud is fully globalized and supports multiple languages. English is installed by default and additional languages can be requested. When additional languages are installed, any user can change the current session locale (language) of
their login session through the Preferences menu. However, string administration tools can change display labels to any
language, regardless of the session locale. For example, the active session locale can be French, but you can use any string
administration tool to enter Korean display label values. As a result, Korean display label values will be seen, even if the
session locale is French. Keep this in mind when making changes to field display labels.

To modify display labels in multiple language refer to the topic, Changing a Specific Field Label.

Changing Display Labels Across Multiple Pages

The User Interface Text Tool, sometimes referred to as the String Editor, administers the vast majority of display labels across
Oracle Sales Cloud. It is designed for bulk display label search and replace. This tool supports all installed languages and can
be used to create and modify locale-specific label values.

Note: You cannot modify list of value fields (also known as choice lists) using the User Interface Text Tool.
Instead, use Setup and Maintenance. In some cases, you can also use Application Composer to modify these
special types of fields.

For example, you want to change the display label Products to Our Products wherever it appears throughout Oracle Sales
Cloud.

To make a large-scale change of a display label value across multiple pages:

1. First, sign in as an application implementation consultant and confirm that you are working in an active sandbox.
2. Using the Navigator, select User Interface Text from the Configuration category.
3. Click Search and Replace.
4. In the Search For and Replace With boxes, enter the word or phrase that you want to change and the corresponding
replacement.

Tip: For optimal results, enter search criteria that produces the smallest possible number of search
results. This makes it simpler to preview all possible matches in the next step. To do this, instead of
entering Products, enter ^Products$. Behind the scenes, the search engine finds label values where
Products is the entire display label value.

5. Click Match Case.
6. Click User Interface Text and Global Menu Label Text to search only the display label category that contains the
vast majority of UI display labels.
   All other check boxes can be deselected.
7. Click Preview Changes to review, modify, and exclude individual occurrences before you save your changes.

Changing Specific Display Labels

Sometimes you will only want to change a specific instance of a field label, rather than the labels of all the fields with the
same name. For instance, the Opportunity page includes multiple instances of the label Account, but you might only want to
change one. If you’re working in an English locale, make this change using Application Composer. If you’re in a non-English
locale, use the User Interface Text tool.

To change an individual field name using Application Composer, look at the object’s fields, then change each one with the
same name to a unique value. For example, if you want to change the Account field and there are four instances of it, then
change each one to something like Account 123, Account 456, and so forth. Then you can look at the object’s UI to identify
the one you want to change. After making the change, update the others to their original names, and make a note in the
Description field for each one to help others in the future.
For a non-English-localized environment, you can’t just use Application Composer to change the field name, because Application Composer is supported only for an English locale. Instead, access the User Interface Text tool after changing each field label to a unique name. In the User Interface Text tool, specify the language you want to change to in the Language list, then enter the unique name of the field you want to change in the Search box and the name you want to change to in the Replace box.

See Changing a Specific Field Label: Explained.

Changing Custom Object Display Labels
For example, you want to change all the display labels for a custom object, Trouble Ticket. In this case, use Application Composer.

Use Application Composer to change a display label value for custom object and custom field display labels. The change is displayed everywhere the original label value was displayed, across all UI channels.

Note, however, that if you use Application Composer to change display labels for standard object and standard object standard fields, the change is not displayed everywhere. This is because Application Composer changes only one instance of a label, while there are typically multiple instances of a label in use for every standard object and standard object standard field. In this case, Application Composer and Page Composer are similar tools: both tools change one and only one instance of a display label. Again, only the User Interface Text tool has the potential to change every instance of a display label.

Application Composer is an English only tool. It reads and writes only to English label data sources. Application Composer reads and writes in English display label data sources only, even if a non-English locale is the active session locale.

Tip: To add non-English display label values for custom objects, custom fields, standard objects, and standard fields, use the User Interface Text tool.

Related Topics
• Changing a Specific Field Label: Explained

Business Process Models: Explained
The application is based on business process models that map out business flows. When you configure and extend your application, for example to add new pages, you can use these models to help you plan. For diagrams of business process models, see Oracle Fusion Business Process Models (1542019.1) on My Oracle Support at https://support.oracle.com.

Business Process Modeling Levels
The business flows are presented in a five-level hierarchy: industry (L0), business process area (L1), business process (L2), activity (L3), and tasks (L4).

• The hierarchy goes from a high-level, conceptual view to a low-level, application-specific view.
• L1 through L3 are business-driven and don’t depend on any specific implementation in the application.
• L4 aligns with specific features and functionality in the application.
In the Application

The application is organized around these hierarchy levels and flows, which puts focus on the activities and tasks that you must perform. Several aspects of the application are influenced by, if not directly based on, the business process modeling levels. For example, the navigation, user interface, and parts of security are influenced by the business process modeling levels.

Enabling the Testing of Your Application Configurations

What's Required for Testing Configurations in the Sandbox

If you are creating configurations for a specific job role or if you are creating your own objects, then you must be provisioned with additional job roles to view and test those configurations in the sandbox. You can enable the testing of both types of configurations using the steps described in this section.

What's Required for Role-Specific Configurations

If you are creating configurations for a specific job role in either Application Composer or Page Composer, then you must assign yourself that same job role to be able to test the configurations in the sandbox. For example, if you are creating your own page layout for the Sales Manager job role, then you must have the Sales Manager job role to view and test the layout. If you later create a different layout for salespersons, then you must deprovision the Sales Manager job role and provision yourself with the Sales Representative job role instead.

What's Required for the Objects You Create

If you are creating your own objects, then you must assign yourself the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. The application automatically generates this object role the first time you create an object in the application. Unless users have this role, they cannot view or test the objects they create.

Setup Overview

1. While signed in as a user with security privileges, such as the setup user or the initial user you received when you signed up with Oracle Sales Cloud, you edit all of the role-provisioning rules for sales administrators and add the required job roles. Here is a summary of the steps:
   a. You search for and open the Manage HCM Role Provisioning Rules task in the Setup and Maintenance work area.
   b. You search for all role-provisioning rules containing the Sales Administrator job role.
   c. For each rule, you add the job roles required for testing. Selecting the Self-requestable option makes it possible for individual users to assign themselves each job role when needed.
   d. If you are creating your own objects, then you must also add the Custom Objects Administration role. You must select both the Self-requestable and the Autoprovision option for this role. This object role is required for all objects you create, so you want to provision it automatically for future to sales administrators.

   For details, see the Enabling Sales Administrators to Test Configurations in the Sandbox topic.

2. Sales administrators, who are resources with the Sales Administrator job role, navigate to the Resource Directory and assign themselves the job roles they need. Setup users, who are not resources, can edit their own user records in the Manage Users work area and assign themselves the roles there.
For details on how resources can assign themselves job roles in the Resource Directory, see the Assigning Yourself an Additional Job Role topic.

Enabling Sales Administrators to Test Configurations in the Sandbox

Modify the provisioning rules to make it possible for sales administrators to assign themselves the job roles they need for testing configurations in the sandbox. For viewing and testing objects they create, sales administrators must have the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. To test job role-specific configurations, they must have the same job role.

Modifying the Provisioning Rules for Sales Administrators

1. Sign in as a setup user or the initial user you received when you signed up with Oracle Sales Cloud.
2. Navigate to the Setup and Maintenance work area.
3. Search for the Manage HCM Role Provisioning Rules task.
4. Click the task name link in the search results.

   The Manage Role Mappings page appears.
5. Search for the role mappings that provision the sales administrators:
   a. In the Search region, click the Role Name list and select the Search link.
   b. In the Search and Select window, enter Sales Administrator in the Role Name field and click Search.
   c. Select the role name and click OK.
   d. Click Search.
6. On the Manage Role Mapping page, click Search.

   The Search Results display the mappings with the Sales Administrator job role.
7. Click the mapping name of each mapping and make the following edits:
   a. In the Associated Roles region, click Add Row (the plus sign icon) and add the job roles required for testing.
   b. For each job role, select the Requestable and the Self-requestable options and deselect Autoprovision. You do not want the job roles assigned to the sales administrators automatically.
   c. If you are creating your own objects, then you must also add the Custom Objects Administration role. The application automatically generates this object role the first time you create an object. For this job role select all of the options: Requestable, Self-requestable, and Autoprovision. All users creating their own objects must have this role.
   d. Click Save and Close.
8. When you have added the job roles to all the provisioning rules, click Done.

Assigning Yourself Additional Job Roles Required for Testing

Sales administrators who are also sales resources can use this procedure to assign themselves the role they need to test role-specific modifications in the sandbox. For example, an administrator testing UI modifications for sales managers, requests the Sales Manager job role or the equivalent customer-defined role. If you are creating your own objects, you can use this procedure to assign yourself the Custom Objects Administration role, if this role is not already assigned to you. The Custom Objects Administration role is required for testing your objects in the sandbox.

Note: You can only assign yourself job roles that are made self-requestable in the role-provisioning rules created by a setup user. A setup user has the privileges to create other users and manage application security.
Assigning Yourself an Additional Job Role

2. Select **View Resource Details** from the **Actions** menu in your record.

The Resource page appears.

3. Select the Roles tab.
4. Click **Add Role**.

The Add Role window appears.

5. Search for the role you want to use for testing by name or partial name, select it, and click **OK**.

For testing objects you created, you must add the Custom Objects Administration role.

\[\text{Note:} \] Available roles include only those that were set up as self-requestable during provisioning rule setup.

The application returns you to the Resource page and displays the requested role in the Roles Requests region.

6. You can remove a role you no longer need for testing by selecting it and clicking **Remove**.
7. Click **Save and Close**.

The new role becomes available for your use in a few minutes, pending the completion of a background process. It displays in the Current Roles region the next time you navigate to this page.
3 Application Changes Life Cycle

Configuration Life Cycle: Explained

All configurations and extensions must be done in a test environment. Typically, this environment contains one or more applications that will be moved to a production environment after you complete and test all configurations and extensions.

While using tools such as Page Composer, you must make application changes in a sandbox. Sandboxes store configurations in Extensible Markup Language files in a separate Oracle Metadata Services repository that's available only when you work in that particular sandbox.
Project managers can monitor and export configurations. Then, users can test the entire environment with all changes, as shown in the following figure.

Runtime Application Changes

Runtime Configuration Workflow: Explained

While using Application Composer and Page Composer to make runtime changes to your application, use sandboxes to save your changes in a segregated environment. For example, before making application changes, suppose you create a sandbox...
named MySandbox, and then make your changes in that sandbox. Now, if others want to see your changes, they can use MySandbox.

> Note: If you have multiple users working on the same sandbox, then conflicts may arise within a sandbox. Hence, users must adhere to the prescribed guidelines to avoid such conflicts.

After you complete your application changes, others can review and approve your changes, and then publish to the test environment.

Note that a flexfield sandbox is for testing only and can’t be published. Instead, you can deploy a flexfield to the test environment using the flexfield UI. To test a flexfield configuration before deploying it to the test environment, deploy it to a flexfield sandbox. The changes that you deploy to a sandbox are isolated from the test environment. Users who make the flexfield sandbox active in their session can only see these changes. After you’re satisfied with the changes in the sandbox, you can deploy the changes to the test environment.

You can also use the Manage Customizations dialog box to:

- View others’ configuration metadata files
- Download others’ configuration metadata files for manually moving them to another environment or diagnosing any issues

Viewing and Diagnosing Runtime Application Changes: Points to Consider

Use the Manage Customizations dialog box to view and diagnose runtime changes made to application pages. Application changes are role-dependent and by default, the Manage Customizations dialog box displays the changes that the signed-in user had performed.

Before you begin viewing application changes, ensure that you have administrative privileges to access the Manage Customizations dialog box. If you’re unable to display the page that contains the changes:

1. Click your user image or name in the global header, and select Manage Customizations from the Administration menu.
2. Use the Search text field on the Manage Customizations dialog box to search for the page, page fragment, or task flow.

You can view the application changes for a user in the Current Context column on the Manage Customizations dialog box. On this dialog box, you can change the page, page fragment, or task flow for which you’re viewing application changes using the Search field.

Developers too may be assigned to specific roles and can view only those application changes that are permitted for the specific roles. However, administrators can view all application changes made at the site level, and for any user, in the All Layers column on the Manage Customizations page. To view application changes made by more than one user, administrators can select multiple users.

Page-Level Changes

To diagnose issues pertaining to application changes, determine whether changes have been applied to a page. Use the Manage Customizations dialog box to determine if page-level changes exist. If a page modification causes problems, such as a user interface component disappears from a page, you can export the application changes and examine the document file.
Oracle Sales Cloud
Extending Sales

Chapter 3
Application Changes Life Cycle

Related Topics

- Context Layers: Explained

Sandbox Manager

Using Sandboxes: Overview

When making changes to Oracle Sales Cloud, use the Sandbox Manager to work within sandboxes, which are separate areas that users can’t see. Test your changes in the sandbox before making your changes available to everyone. Read this chapter to understand how best to work with sandboxes, safely and efficiently.

This chapter covers:

- How to create and activate sandboxes
- How to best work in a sandbox when others on your team may also be testing application changes in their own sandboxes
- Which types of application changes you cannot do inside a sandbox
- How to set up, manage, and publish sandboxes

Maintain sandboxes using the Sandbox Manager. In the global region, expand the Settings and Actions menu, which is available when you click your user image or name. Then select Manage Sandboxes, under the Administration subheading.

Sandboxes: Explained

Today’s dynamic business landscape demands fast responses from companies to address both customer and market needs, typically requiring several different teams to work simultaneously on application changes while sharing the same data model and configuration starting point. Oracle Sales Cloud use sandboxes to allow companies to meet these requirements. Sandboxes let companies avoid the risk of conflicts between teams working in parallel, and give administrators the ability to test all application changes before their users ever see them.

Sandboxes in Oracle Public Cloud Services provide robust, ready to use functionality to help isolate and control application change efforts without affecting other users’ configuration environments, or the production environment. You can read all about sandboxes in the Oracle Public Cloud Services Extensibility Guide for Business Analysts. Or, review this guide to learn how to modify Oracle Sales Cloud, specifically, using sandboxes.

Sandboxes let users make changes isolated from the application in the mainline code as well as from other sandboxes. The mainline code is the source of data and definitions used at the time of creating a new sandbox.

Business analysts can implement and test application changes in a sandbox and, once satisfied, publish them to the mainline code. When making changes in a sandbox, your application changes won’t be available to any other sandbox or to any application in the mainline code until you have published your sandbox. When publishing a sandbox, the included application changes replace the mainline code application’s existing configuration.

Note: Never make your application changes directly in the mainline code. Instead, always use sandboxes whenever possible. You must use a sandbox even when you use the Oracle Fusion Functional Setup Manager.

Oracle
Composers
To modify applications within Oracle Sales Cloud, you can use these composers:

- Application Composer: Modify pages, business objects, and all the artifacts that support them (such as fields, pages, buttons and links, security, server scripts, and saved searches). Extend Oracle Sales Cloud by creating completely new business objects and artifacts.

- Page Composer: Modify pages.


>Note: To complete most tasks in Application Composer, you must be in an active sandbox.

Application Changes Affect Metadata Services and the Database Layer
At a technical level, your application changes affect two major areas: the Metadata Services (MDS) repository and the database layer.

First, all changes result in the creation or updating of many files within the MDS repository. Your application changes are stored as XML files in the repository, segregated by sandbox.

Additionally, as custom objects and fields are created, their definitions are allocated to generic placeholders that already exist as tables or columns in the database.

Sandboxes handle metadata changes made to the data stored in the Metadata Services (MDS) repository.

Sandbox Usage
Sandboxes typically have one of two purposes:

- Test-Only: Users perform all application changes using the test-only sandbox. Changes made in the test-only sandbox must never be published to the mainline code.

- Publish: Once satisfied with the application changes made in the test-only sandbox, users replicate their changes in this sandbox, and then publish them to the mainline code. This sandbox type is also known as the integration sandbox, because teams working in parallel use this sandbox as the final staging point before publication to the mainline code.

>Note: Before each patch or upgrade, publish or delete your sandboxes. Outdated sandboxes might not be publishable due to potential conflicts. If your work is incomplete, you must restart with a new sandbox.

Sandbox Manager
You can maintain sandboxes using the Sandbox Manager:

- Create a sandbox
- Activate a sandbox
- Delete a sandbox
- Publish a sandbox
- View available or published sandboxes
Managing Configurations Using Sandboxes: Explained

You can apply different types of configurations to an application. For example, you can apply changes to an application's metadata stored in the metadata services repository or changes related to data security of the application. All such configurations are stored in sandboxes and are validated before applying them to an application.

Types of Configurations in Sandboxes

Sandboxes can contain the following types of configurations:

- Metadata changes - These changes (such as non-flexfield user interface (UI) page changes) are captured in a metadata sandbox.
- Data security changes - These changes are additionally captured in a data security enabled sandbox.
- Changes in the generated flexfields business components - These changes are captured in a flexfield that's deployed as a single flexfield sandbox.

Once you’re ready to make sandbox changes available in the mainline metadata, either publish the metadata or data security sandbox, or deploy the flexfield. You can download only metadata and data security sandboxes as a sandbox file for import to another application instance.

The following table lists the differences among the types of sandboxes.

<table>
<thead>
<tr>
<th>Type of Changes</th>
<th>Type of Sandbox</th>
<th>Method for Making Changes Available in Mainline Metadata</th>
<th>Downloadable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>Sandbox</td>
<td>Publish sandbox</td>
<td>Yes</td>
</tr>
<tr>
<td>Data security</td>
<td>Sandbox enabled for data security changes</td>
<td>Publish sandbox</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexfield</td>
<td>Flexfield deployed as a flexfield-enabled sandbox</td>
<td>Deploy flexfield</td>
<td>No</td>
</tr>
</tbody>
</table>

Only one sandbox can be active at a time. All changes made in an active sandbox are captured in that sandbox.

Environment

To make application changes in runtime, you must first create a sandbox and then use tools, such as Page Composer to make the changes. These changes remain within the sandbox and don’t affect the mainline metadata. You can test and validate the changes by publishing the sandbox to a test environment. After testing the application, you can move to the production environment. The application changes made in the sandbox will be migrated to the production environment and will be available to the users.

Don’t make application changes directly in the mainline metadata. Make all application changes in the sandbox first. When you make changes to an application at runtime in a sandbox, you isolate the changes from the mainline metadata. After completing the changes in the sandbox, verify them. When you’re ready to save the changes, publish the metadata or security-enabled sandbox to the mainline metadata.

When you create a sandbox, you can see the information pertaining to only the existing application changes in the current mainline metadata. For example, suppose you make an application change in a sandbox and publish it. Then, on creating
another sandbox for the next change, you will see the previous change in the new sandbox because that change exists in the current mainline metadata.

Flexfield sandboxes are for testing only and can’t be published. Make flexfield configurations that are stored in a database. Then, deploy those configurations to a sandbox to see the resulting deployment artifacts in a sandbox environment. Flexfields are deployed directly to the mainline metadata using the flexfield user interface.

Tools

You can use several runtime configuration tools to make application changes. For example, you can modify objects and pages using Page Composer, which uses sandbox manager. Oracle Business Process Composer and Oracle SOA Composer are also tools used for making runtime application changes, but they don’t use sandbox manager. They have their own mechanisms for handling application changes.

Managing a Flexfield Sandbox

To create a flexfield-enabled sandbox, deploy a flexfield to a sandbox using the Manage Flexfield task flow. The flexfield sandbox gets its name from the flexfield you deploy. You can’t test two flexfields in the same sandbox. After deploying a flexfield as a sandbox, sign out and sign in again to view how the sandbox runtime reflects the flexfield changes, such as new segments. You can redeploy the same flexfield to the same sandbox repeatedly as you make incremental changes to the flexfield setup. A flexfield sandbox can’t be published. So, when the flexfield is deployed to the mainline metadata, any page changes or data security in the flexfield sandbox can’t reach the mainline metadata. If you’re entitled to do so, manage flexfield-enabled sandboxes in the Sandbox Manager.

Related Topics

- Deploying a Flexfield to a Sandbox: Points to Consider

Using Sandboxes: Points to Consider

In the runtime configuration workflow, use sandboxes to isolate the changes from the mainline metadata for testing and validating. After you’re satisfied with the changes, you can publish the changes back to the mainline metadata.

The testing sandboxes are never published and therefore produce no concurrency conflicts between sandboxes. You can have several testing sandboxes at the same time. But if you have multiple users working on the same testing sandbox, then they must adhere to the prescribed guidelines.
Application changes in the sandboxes that are published are merged back to the mainline metadata. The following figure illustrates the two types of sandboxes and their relationship to the mainline metadata.

Working with a Single Sandbox

When multiple users are making changes in an application using the same sandbox at the same time, conflicts within a sandbox may arise. This conflict may arise because multiple users attempt to modify the same artifact or make application changes that indirectly affect other shared files. For example:

- A direct conflict arises when different users attempt to modify the same page, fragment, or metadata file within the same layer.
- An indirect conflict arises when two users, each creating a different object, save their changes at the same time. This conflict occurs in the metadata file that tracks which new objects both users created while saving their changes.

Conflicts may also arise when users edit a shared artifact, such as when a user performs an operation that adds or edits a translatable string. For example, say:

- One user edits a field’s display label or help text, or a validation rule’s error message. Whereas, another user performs an operation at the same time that similarly affects translatable strings.
- Two or more users are working in Navigator menus that are shared across applications. Whenever a conflict arises among users, the application displays concurrency warning messages.
Whether the sandbox is meant for testing or production, if multiple users work with a single sandbox, follow these guidelines to avoid conflicts:

- 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 must not update object1. Suppose both modifications involve changes to translatable strings, and the users save changes to separate objects at the same time. Even then, a conflict can occur in the resource bundle that stores the translatable strings.

- If multiple users update the same artifact (same object or same underlying frequently modified file) concurrently, then they will get a concurrent update error. The second user’s updates won’t be saved (the Save button will be disabled), and one of the users will have to cancel and try again.

- All users using the same sandbox should have the same application role. Users with different roles might not be able to see all content created by other users within the sandbox.

**Working with Multiple Sandboxes**

Multiple sandboxes are used when configurations are stored in testing as well as production sandboxes. Say, after you create a sandbox, a concurrent change is made in the mainline metadata. Now, when you attempt to publish that sandbox, the application detects such conflicts at publication time, and you get error messages.

**Note:** When you publish your sandbox, you may get a message showing a conflict on oracle/apps/menu/fnd/appcore/dataSecurity/dataSecurityService/mds/DSMO.xml. This message indicates that the security changes that you made in your sandbox conflict with other security changes in the mainline metadata. Delete the sandbox and recreate your changes in a new sandbox.

If multiple users are permitted to work in multiple sandboxes at the same time, follow these guidelines to avoid conflicts:

- Any number of test-only sandboxes can operate 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 aren’t published, cause no conflicts with each other. Be aware, however, that all modifications will be lost when the sandboxes are destroyed.

- For sandboxes that aren’t for test-only and will be published, users can use multiple concurrent sandboxes only if they operate on mutually exclusive artifacts. For example, you can have:
  - One sandbox that contains a page that a user is modifying to add a task flow
  - Another sandbox that contains a different page from a different application

However, some objects might still share underlying artifacts, and thus it’s not always obvious if two objects are truly mutually exclusive. Thus, proceed with caution when using multiple concurrent sandboxes that will be published. It’s still possible that a conflict could occur, which would require the deletion of one or more sandboxes.

- Suppose the users update an artifact in both, the mainline metadata and in one sandbox, or in two different sandboxes. Now, when you publish the sandbox, the application detects such conflicts and you get an error message. At this point, cancel publishing the sandbox to avoid overwriting previous changes.

**Note:** For a sandbox that contains configurations pertaining to ADF Business Components, sign out and sign in again after switching in or out of this sandbox. This process ensures avoiding any inconsistencies between the runtime caches and the ADF Business Components definitions.
Sandboxes: How They Work with Application Changes and Features

This topic covers the considerations you must keep in mind when working in a sandbox and using certain features and composers, such as Application Composer and Page Composer.

Lookup Types and Values

Lookup types and lookup values are considered seed data, and are not stored inside the Oracle Metadata Services (MDS) repository. Accordingly, any lookup types or lookup values that you create as part of an application change are retained in the database, even after a sandbox is deleted.

Enterprise Scheduler Service

ESS is not aware of sandboxes and will operate only in the mainline metadata. When you are in a sandbox and submit an ESS job, the ESS job will run outside of sandbox.

Reports and Custom Fields

You can use the Oracle Business Intelligence (BI) Composer to build custom reports. During report creation, you select a report subject area as the basis for your new report. A report subject area contains a set of objects and fields that represent information about the areas of an organization’s business. Many report subject areas are already available to you as part of Oracle Public Cloud Services.

Note that you can also create custom subject areas, which are report subject areas that you build using Application Composer. To create a custom subject area, however, you must be in the mainline metadata application; you cannot be in a sandbox. Therefore, if you want your report to include custom fields or objects (always created inside a sandbox), you must first publish your sandbox. Only after publication can you create a custom subject area that includes the custom fields or objects that you want to later report on.

Web Services (and Object Workflows)

Web services do not reflect sandbox changes such as custom fields or objects until the sandbox is published. Consequently, features that depend on Web services to work will not gain access to the custom fields or objects until the sandbox is published.

For example, when working with object workflows, you can create a custom field and define a workflow condition using that field. While working in a sandbox, however, you cannot reference the custom field in the workflow actions because workflow actions rely on Web services to get field values. Therefore, you must first publish the sandbox in which you created your custom field. Only after publishing the sandbox can you then update the object workflow’s condition using your custom field.

Import and Export of Custom Objects

To support the importing and exporting of the custom objects that you created with Application Composer, you must first generate the object artifacts required for both file-based import and bulk export. Note that this task is not supported from within a sandbox, and can only be completed in the mainline metadata application.

Using Sandboxes: Explained

Maintain sandboxes using the Sandbox Manager. You can open Sandbox Manager by clicking your user image or name in the global header and selecting Manage Sandboxes, under the Administration subheading.
Use the Sandbox Manager to:

- Create sandboxes
- Activate sandboxes
- Exit a sandbox and return to mainline metadata
- Review a list of available or published sandboxes
- View sandbox details
- Publish sandboxes
- Delete sandboxes

**Creating a Sandbox**

Using the Sandbox Manager, create a new sandbox by using the Actions menu option, or by clicking the New button.

When creating multiple sandboxes, create one for testing only which you will never publish. Also, create a single integration sandbox that you do intend to eventually publish.

- **Private sandbox** - Testing and prototyping only. Never publish. Delete when finished, or after its related integration sandbox has been published.
- **Integration sandbox** - Testing and validations with the intent to publish. Ensure only one administrator user works in this sandbox at a time.

Coordinate with other administrator users to manually migrate (reenter) approved configurations from a private sandbox into the integration sandbox. To avoid confusion, establish naming conventions such as `rjones4_19nopub`, `mhoope4_19nopub`, and `integrationsandbox4_19topub`. The date indicates when a sandbox was derived from the application in the mainline metadata. You can also check the sandbox creation date and time using the Sandbox Manager.

**Activating a Sandbox**

After creating a new sandbox, you must next activate it to be able to use it. To activate a sandbox, select the sandbox and then click the Set as Active button. Only one sandbox can be active at a time.

Once a sandbox is active for your session, the sandbox name is displayed in the global header.

After activating a sandbox, you must always sign out from Oracle Public Cloud Services and sign in again. This helps you to avoid conflicts by ensuring that the cache is cleared.

Note that if you sign out and sign in again, your sandbox remains active. A sandbox remains active until you exit the sandbox, publish the sandbox, delete the sandbox, or set another sandbox as active.

**Exiting a Sandbox and Returning to the Mainline Metadata**

To exit from the current sandbox session, hover over the sandbox name in the global header and then click Exit Sandbox.

The sandbox session is closed and you are returned to the application in the mainline metadata. After exiting a sandbox session, you must always sign out from Oracle Public Cloud Services and sign in again. This helps you to avoid conflicts by ensuring that the cache is cleared.

⚠️ **Caution:** Once back in the application in the mainline metadata, avoid making application changes using Application Composer. To start making application changes again, use the Sandbox Manager to set a new sandbox as active.
Viewing Sandbox Details

After you’re in a sandbox, you can view details about it using the Sandbox Details window.

To view the Sandbox Details window, click the name of the active sandbox at the top of the window, then click More.

⚠️ **Caution:** In the Sandbox Details window, it is possible to select individual files within the sandbox and delete them. You must not do this, because it can cause problems with your sandbox.

Publishing a Sandbox

Completed application changes created within a test-only sandbox and then replicated to an integration sandbox must be published to be available to other users in the application in the mainline metadata. Always publish application changes from the integration sandbox only.

Before each patch or upgrade, you should publish or delete your sandboxes. Outdated sandboxes might not be publishable due to potential conflicts. If your work is incomplete, you should start fresh with a new sandbox.

Note that there is no standard mechanism to roll back changes that have already been published to the mainline metadata.

To publish a sandbox, select the sandbox and then click the Publish button.

After you publish a sandbox, the sandbox session is closed and the sandbox is no longer active. Be sure to delete your test-only sandboxes, and then create new sandboxes (including a new integration sandbox) for new application changes.

Deleting a Sandbox

Deleting sandboxes cleans up the Metadata Services (MDS) repository and database layers.

Once you have tested your application changes, you then move those changes to the integration sandbox. After you publish your integration sandbox, you must delete all test-only sandboxes, and then create and work in entirely brand new sandboxes, including a new integration sandbox. You can delete only nonpublished sandboxes that are not active.

⚠️ **Caution:** Although you might delete a sandbox, transactional data for custom objects is retained because transactional data is stored outside of the MDS. Suppose a custom object named D1 is created in a sandbox, three rows of transactional data were entered through its work area at run time, and then the sandbox is deleted. The three rows of transactional data are retained, although not visible to users unless a new custom object is created with the exact same name (D1) with the same fields in the same order. In this case, the data might be exposed once again. Deletion of partial content of a sandbox is risky; it is recommended that you do not use this option.

Supported Sandbox Manager Operations: Explained

This topic contains operations that are supported by sandboxes.

Supported Sandbox Operations

Sandboxes support the following operations:

- Create
- Activate: Only one sandbox can be active at a time.
• Delete: Delete a sandbox only when the sandbox is no longer needed, the sandbox is outdated, or its related integration sandbox has been published to the mainline metadata.

• Publish: Publish a sandbox using extreme caution. Once a sandbox has been published, all existing sandboxes derived from the same mainline metadata are now invalid. There is no rollback operation for published sandboxes.

• Download All: Coordinate this operation with the main administrator user, before publishing a sandbox, as a way of performing a backup of current sandbox changes. This backup can be shared with Oracle Support Services, should you encounter a scenario that you cannot resolve.

• Exit: Exit the sandbox.

You must first sign out and then sign in back into Oracle Sales Cloud when you perform the following operations:

• Activating a sandbox.
• Exiting a sandbox.
• Publishing a sandbox.

**Importing Sandboxes**

Don’t import sandboxes. This operation is reserved for Oracle internal development only.

**Setting Up Sandboxes: Procedure**

To make changes to the application artifacts, you must first store them in an active sandbox. You can either create a sandbox or select an existing sandbox, and designate it as an active sandbox. The active sandbox holds the context for all the changes. The sandbox uses a database to store the actual changes. After testing your changes, you can publish the sandbox, or deploy the flexfield, and the changes are merged into the mainline metadata. Eventually, the sandbox is archived.

The following procedure is for setting up non-flexfield sandboxes. For flexfields, use the Manage Descriptive Flexfields task or the Manage Extensible Flexfields task.

⚠️ **Caution:** Don’t import or delete metadata files. These operations modify sandbox contents and could cause issues with the sandbox functionality.

To create a sandbox and set it up:

1. Click your user image or name in the global header, and select Manage Sandboxes from the Administration menu.
2. Use the Manage Sandboxes dialog box to create a sandbox.
3. Click Save and Close.
4. On the Manage Sandboxes dialog box, select the new sandbox or an existing one, and click Set as Active. The sandbox is designated as the active sandbox.
5. Close the Manage Sandboxes dialog box.

**Publishing Sandboxes: Procedure**

After completing the application changes in the sandbox, publish them to make them available in the application.

**Prerequisites**

Before publishing the application changes, test or validate the changes at runtime using test systems and any combination of the validation setups.
If there are changes to the mainline metadata from another source and you publish your sandbox data, then the mainline metadata isn’t overwritten. However, if you get error messages notifying about conflicts, then you must fix the conflicts before publishing.

Publishing Sandboxes
To publish a sandbox:

1. Click your user image or name in the global header, and select Manage Sandboxes from the Administration menu.
2. On the Manage Sandboxes dialog box, select the sandbox and click Publish. The Publish confirmation message box appears.
3. Click Yes. The sandbox is published to the mainline metadata.
4. Close the Manage Sandboxes dialog box.

Multiple Sandbox User Conflicts: Explained
This topic explains the situations under which conflicts might occur when multiple users are working in sandboxes, and provides guidelines on how you can avoid such conflicts.

Application changes are stored as XML files in the Metadata Services (MDS) repository and are segregated by sandboxes. When you modify an application artifact, your changes typically affect multiple metadata files directly or indirectly; therefore, when multiple users are working in the same sandbox or with different sandboxes intended for publishing, conflicts might happen.

Multiple Users in a Sandbox
When multiple users work in a sandbox, one user might overwrite the changes performed by the other as follows:

- Directly - by changing the same artifact object.
- Indirectly - by updating metadata files shared between different artifacts.

If changes on the same object are saved at different times, then the last saved change replaces the other changes. For example: if user A and user B are working in a sandbox and both are updating the same object. Call their changes change A and change B, respectively. When both users save their changes at the same time, only one of the changes is saved, say change A is saved. Change B is not lost, but when user B saves change B again, change A is overwritten.

Multiple Integration Sandboxes
When multiple sandboxes exist for publication, one user might overwrite the changes made by the other user when sandbox is published. The resulting mainline metadata configuration is always from the last published sandbox.

How to Avoid Conflicts When Working in Sandboxes
Use these guidelines to avoid conflicts when working in sandboxes:

- Use a distinct user name for each administrative user.
  Within the Customer Relationship Management Application Administrator duty, ensure that every administrative user has a distinct user name. Don’t share your user names to perform application changes.
- Create a single integration sandbox at a time.
  Never create more than one integration sandbox at a time. Only create another integration sandbox once the previous sandbox is published.
• Enforce a single user per sandbox rule except when integrating sandboxes.

In most cases, users must not be in the same sandbox at the same time. Ensure that only a single user is in a sandbox at a time. (You must manually enforce this rule.)

The application doesn’t prevent multiple users from sharing a single sandbox. If you want to isolate your own implementation project in a single sandbox to test or modify it, you should ensure that no other users are sharing the sandbox. However, when multiple users are ready to merge all their changes into a single integrated sandbox, they can all share the sandbox to test the integration. In this case, all the users must be aware that the sandbox is shared, and coordinate their integration testing after all of their changes have been made in the sandbox.

FAQs for Using Sandboxes

When do I publish a sandbox?

You can publish a sandbox after you have tested and verified that the application changes done in that sandbox are ready to be moved to the mainline metadata.

You must test the following configurations outside a sandbox:

• Import/Export
• Web services
• Custom subject area creation
• Object workflow
• E-mail templates

How frequently can I publish a sandbox?

Integration sandboxes are typically published once a week. Publishing integration sandboxes less frequently than once a week isn’t recommended.

When you publish an integration sandbox, all private sandboxes are invalid because the label in the mainline metadata application has changed. If you made changes to private sandboxes that you want to retain, then document those changes and then delete all the private sandboxes.

How can I manage server exceptions while publishing a sandbox?

When publishing a sandbox, a server exception may be encountered. Follow these guidelines, depending on the exception error that you encounter:

• ProfileMO.xml Error: If you encounter a message showing a conflict on /oracle/apps/fnd/applcore/profiles/profileService/mds/ProfileMO.xml when you publish your sandbox, you can ignore this message and continue to publish the sandbox.

• Log an Oracle Technical Support Request with the Incident Number found on the error message and the name of the sandbox.
• Create a new sandbox, activate it, and resume your work.

Can I delete a sandbox?

Yes. You can delete sandboxes. However, you can delete only those that aren’t published. Before you delete a sandbox, you must first confirm that the sandbox isn’t active.

⚠️ **Caution:** Deletion of partial content of a sandbox is risky. It’s recommended that you don’t use this option.

After you have tested your application changes, you must move those changes to the integration sandbox. Publish your integration sandbox and then delete all the test-only sandboxes. You can then create and work in new sandboxes, including a new integration sandbox.

What Application Composer tasks are available only within a sandbox?

Most Application Composer tasks require you to be in a sandbox. For example, these menu items are available to you only if you’re in an active sandbox session.

- Objects
  - Custom Objects
  - Standard Objects
- Common Setup
  - Relationships
  - Role Security
  - Object Workflows
  - Global Functions
  - Run Time Messages
  - Mobile Application Setup
  - Outlook Setup
  - Personalization
  - Web Services
  - Metadata Manager

These menu items are the exceptions. They’re available only in a sandbox-free session.

- Custom Subject Areas
- E-Mail Templates
- Import and Export
- Business Processes
Moving Application Changes

Using Configuration Migration to Move Configurations: Points to Consider

Use the Configuration Set Migration page to create a set of all configurations and extensions made to an environment. Then, download the configuration set and upload it into another environment. The configuration set includes configurations across all product families. To open the Configuration Set Migration page, select Configuration > Migration from the Navigator menu.

Contents of the Configuration Set

The configuration set includes:

- Application changes done using Application Composer, except the following changes:
  - Object artifacts that were generated from the Import and Export page in Application Composer to make extensions available for importing and exporting
  - User names and passwords for secured SOAP web service connections
  - The enabled attachment feature for custom objects

- Changes made to application artifacts using the following tools:
  - Page Composer
  - Appearance
  - Structure
  - User Interface Text
  - Page Integration

- Changes in the following artifacts of the Applications Core Setup application:
  - Messages
  - Lookups
  - Data security
  - Descriptive, extensible, and key flexfields, and value sets
  - Attachment categories and metadata
  - Deep Links

- Changes in Reports and Analytics such as regeneration of SOAP services, including user-defined attributes
- Changes in CRM email templates created in Application Composer
- Changes done using the Manage Oracle Social Network Objects task
- Changes in functional security settings made in Application Composer, including functional privileges that control access to custom objects
Note: Enterprise roles, new duty roles, and role hierarchy changes, which are made directly in Security Console aren't migrated. If functional security associated with roles in the source are migrated to a target instance, and the corresponding role doesn’t exist in the target instance, an error will occur on import.

The configuration set may also include:

- Changes in Enterprise Scheduler Service (ESS) module, such as:
  - Job definitions
  - Job sets
  - Job schedules
  - Incompatibilities
  - Work shifts
  - Work assignments
  - Work assignment schedules
  - Job request parameters

- Changes in Service Oriented Architecture (SOA) artifacts such as configurations done using SOA Composer

- Changes done using Oracle Business Intelligence Enterprise Edition, including but not limited to:
  - Oracle Business Intelligence Answers
  - Oracle Business Intelligence Delivers
  - Business Intelligence Composer
  - Dashboard Builder
  - Oracle Business Intelligence Publisher

Note: You can move the configurations done using the business intelligence tools only if the Business Intelligence in Configuration Set Migration Disabled profile option is set to No.

The configuration set doesn’t include personalizations.

While an upload or restore activity processes Presentation Services changes, the following can occur:

- Reports that were submitted by Oracle Enterprise Scheduler to Oracle Business Intelligence Publisher and were scheduled to execute during the process, will fail.
- The Reports and Analytics pane may not display.
- Oracle Business Intelligence Publisher reports may not display on Oracle Business Intelligence Presentation Services analyses or dashboard pages.
- Users may not be able to access Oracle Business Intelligence Enterprise Edition features, such as:
  - Oracle Business Intelligence Answers
  - Oracle Business Intelligence Delivers
  - Business Intelligence Composer
  - Oracle Business Intelligence Interactive Dashboards
Note: To prevent including in-progress configurations in the configuration set, make your changes in a sandbox. The configuration set doesn’t include changes from a sandbox until the sandbox is published.

You can use the Configuration Set Migration page to move configurations and extensions from any source environment to any target environment. However, you must always make configurations and extensions in a test environment. Then use the Configuration Set Migration page to move these configurations to a production environment. As configuration set migration doesn’t provide a merge capability, never configure or add artifacts in a production environment. When you import a configuration set, the rows in the database that are not preconfigured are updated if a matching record exists. Otherwise a record is inserted.

The configuration set doesn’t include all deletions. For example, the set does not include the removal of a configuration document using the Manage Customizations dialog box. After you import a configuration set into the target environment, you must examine the environment for any deletions that you must make manually. Similarly, the configuration set does not include roles or role hierarchy changes. Changes made to Security Console have to be manually updated in the target environment.

Related Topics

• Importing and Exporting Custom Objects: Explained

Migrating Configurations from Test to Production Environment

Video

Watch: This tutorial shows you how to copy configurations you created in the test environment to your production environment. The content of this video is also covered in text topics.

Procedure

Create a configuration set to move configurations across all product families from the test environment, which is the source environment to the production environment, which is the target. You can export all configurations, such as those stored in Oracle Metadata Services repository, JEDI, CRM, and BI using the Outgoing tab of the Configuration Set Migration page. You can then import them to the target environment using the Incoming tab. You can use a configuration set to move configurations in a batch instead of moving them one by one. Each time you run the migration process in an environment, it copies not just the changes that you have recently made, but all of the configurations published in the environment.

Prerequisites

Before creating a configuration set, ensure that:

• The source and target application environments are of the same release and the same standard and one-off patches are applied to both environments.

• All Page Composer configurations made in sandboxes are complete before they’re published. Before starting the export process, you must publish all complete configurations. All configurations and extensions made using the Structure page, the Manage Standard Lookups task, and Security Console, are complete.

• To move content created using Oracle Business Intelligence Enterprise Edition features, the Business Intelligence in Configuration Set Migration Disabled profile option is set to No in source and target environments. To view this profile option, open the Manage Administrator Profile Values task in the Setup and Maintenance work area.
You have been granted the following privileges, which enable you to access the Configuration Set Migration page:
  - Manage Outgoing Configuration Set
  - Manage Incoming Configuration Set
Contact your security administrator for details.

You never make changes in the target or production environment while applying configurations.

**Note**: If you make changes to the production environment in emergency circumstances, you must make the same changes to the test environment. Making the changes to the test environment ensures that these changes are included in the next configuration migration.

You don’t perform configurations in the source environment during the export process.

Creating Configuration Sets
To create configuration sets:

1. In the source environment, select **Configuration > Migration** from the Navigator menu.
2. From the Outgoing tab of the Configuration Set Migration page, click **Create Configuration Set**.
   
   **Tip**: If the **Delete** button appears for an existing configuration set, click the button. This action removes the temporary files that are on the server from the previous configuration set creation. You can’t create a configuration set until the previous set has been deleted.

3. Provide a name for the configuration set.
4. Optionally, type a description of the set.
5. Select the content you want to migrate.

   **Note**: The Industry solution extensions module is for Oracle’s internal use only and has no impact on CSM operations.

6. Click **Save and Close**.
7. Click **Refresh** periodically to see the current status of the set creation because creating a configuration set can take a few minutes. You can click **Log** to review the process log, which provides more details about the configurations that are being compressed. If the process terminated with a status of Error, the log would provide you information about the configurations that failed to compress. The process runs asynchronously, so you can close the page and return to it later. Eventually, the status changes to Ready for Download, which means that the configurations are ready for you to download. Before downloading the configurations, you can click **Content Read Me** to download the Readme file listing all the configurations you exported.

8. Click **Download**, and specify the name and location for the file, and click **Save**. Ensure that the downloaded file is a JAR file.
9. After you download the file on your local file system, click **Delete** to remove the temporary files that were created on the server.

Applying Configuration Sets
After you apply configurations, end users must sign out and sign in again to see the changes. Hence, apply configurations when a few people are signed into the environment. To apply configurations to the target environment, follow these steps:

1. Open the Configuration Set Migration page in the target environment.
2. Click the **Incoming** tab.
3. Click **Browse**, specify the name and location of the configuration set file, and click **Open**.

   If the Browse button on the Incoming page appears disabled, click the **Outgoing** tab, and click **Delete** to remove the previously uploaded configuration set from the environment. This enables the Browse button.

4. When the status for the configuration set on the Incoming page is Ready to Apply Configurations, click **Apply**.

5. Periodically, click **Refresh** to view the current status of the Apply action. You can review the process log, if required.

   The process runs asynchronously, so you can close the page and return to it later. When the configurations are applied, the status is displayed as Applied and Deleted.

   If problems occur during an Apply action, log a service request using My Oracle Support at https://support.oracle.com.

6. Access the target environment and examine the environment for any deletions that you must manually make.

7. Deploy all flexfields that display a Patched status.

8. Perform the following steps to send the new and updated social network definitions to the social network server:

   a. In the Setup and Maintenance work area, open the Manage Oracle Social Network Objects task.

   b. As part of the applying configurations process, some objects are created or updated. If the Enabled value of such an object is anything other than No, trigger the process of sending its definition to the social network server. You can do it by disabling the object and enabling it again with its original status. For example, if the Enabled value is Manual, then:

      i. Disable the object.
      ii. Enable the object, and select the value, Manual.
      iii. Click **OK** and save the changes.

9. Manually migrate all business processes created in the source environment to the target environment.

10. After applying configurations, perform functional testing to verify the changes. Suppose testing exposes problems with the configurations, such as importing more than you intended, or the changes weren't what you expected. In such cases:

    a. Open the configuration set in the Incoming tab of the Configuration Set Migration page.

    b. Click **Restore** to revert to the state before the configuration set was applied.

       Skip the next step in such cases.

   You can view the process log to monitor the progress of the download or the applying process. This process takes approximately 15 minutes. If it takes any longer and you don’t see any progress, click **Refresh**. You can either let the server take its time and click **Continue**, or click **Restart** to restart the export process.

Following are some important points regarding configuration import and export:

- After an environment upgrade, any previous imports which were performed in an earlier release can’t be reverted. However if a new import is submitted in the upgraded instance, then the most recent import can be reversed.
- Lookup values for lookup fields that exist in both source and target aren’t overwritten during the configuration import. The lookup values from source are added to the target and all the lookup values coexist for the same field. For example, the Status field in its source environment has values, Open and Closed. In the target environment this field has values, Yes and No. After the import, the Status field in the target environment has values, Open, Closed, Yes, and No.
- After importing, perform the following steps in the target environment to send the new and updated social network definitions to the social network server.

   i. In the Setup and Maintenance work area, open the Oracle Social Network Objects task.
ii. On the Oracle Social Network Objects page, click **Synchronize** to synchronize a selected object, or click **Synchronize All** to synchronize all objects together.

- During configuration import, the data security privileges aren’t automatically revoked in the target environment. For example, say a specific privilege is granted in the target environment, but the corresponding privilege doesn’t exist in the source environment. During import, the privilege in the target environment won’t be automatically revoked. To address this issue manually, add such a privilege to the source environment and revoke it. The revoke action is picked up as a configuration instance during the configuration import process and applied to the target environment.
- You can create reports directly in the target environment. However, ensure that you create the reports and reference them to subject areas that were created. Don’t create the subject areas directly in the target environment.
- You can initiate configuration export and import tasks only from the mainline metadata. If you initiate configuration from a sandbox, the process doesn’t execute.

⚠️ **Caution:** All user personalizations that are performed after a configuration set is applied are lost when you perform a restore action on that configuration set.

11. Broadcast information to the users that they must sign out and sign in to view the most recent changes.

**Related Topics**

- Importing and Exporting Custom Objects: Explained

## Exporting and Moving Configurations: Points to Consider

Configurations are stored in XML files. You can use these XML files to export configurations for the following reasons:

- To move configurations and extensions to another environment, such as the production environment.
- To diagnose issues noticed in the test environment.
- To send files to your help desk for further diagnosing.

The following table lists the tools to use to export and move configurations and extensions.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Tools to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move all configurations to another application environment.</td>
<td>Configuration Set Migration.</td>
</tr>
<tr>
<td>Move only descriptive flexfield configurations to another application environment.</td>
<td>Setup and Maintenance work area. Moves configurations for a specified module.</td>
</tr>
<tr>
<td></td>
<td>To move configurations for all modules, use Configuration Set Migration.</td>
</tr>
<tr>
<td>Move only extensible flexfield configurations to another application environment.</td>
<td>Setup and Maintenance work area. Moves configurations for a specified module.</td>
</tr>
<tr>
<td></td>
<td>To move configurations for all modules, use Configuration Set Migration.</td>
</tr>
<tr>
<td>Move only value set configurations to another application environment.</td>
<td>Setup and Maintenance work area. Moves configurations for a specified module.</td>
</tr>
<tr>
<td></td>
<td>To move configurations for all modules, use Configuration Set Migration.</td>
</tr>
</tbody>
</table>
### Tasks

<table>
<thead>
<tr>
<th>Move only lookups to application environment.</th>
<th>Setup and Maintenance work area. Move application standard lookups, application common lookups, or both.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move only data security policies to another application environment.</td>
<td>Setup and Maintenance work area. It doesn’t move Oracle Fusion Human Capital Management roles.</td>
</tr>
<tr>
<td>Export configurations to a file to help diagnose an issue.</td>
<td>Manage Customizations dialog box.</td>
</tr>
</tbody>
</table>

### Downloading Configurations

Use the Manage Customizations dialog box to download configuration files for a given page. You can also upload the files into the Oracle Metadata Services repository using the same dialog box. To open Manage Customizations dialog box, click your user name in the global header, and select Manage Customizations from the Administration menu. You can use these files for diagnosing configuration issues.

You can also download all configurations of a page for all layers (AllCustomization.zip) using the Download Configuration for All Layers link. This link is located at the bottom of the Manage Customizations dialog box. The AllCustomization.zip file contains all the customization XML files for the page. However, you can't upload this file anywhere.

### Downloading Configuration Set Reports

After exporting configurations, you can view and download a configuration set report that contains a list of all configurations available in a configuration set. To do so, click Content Read Me from the Outgoing tab of the Configuration Set Migration page. To open the Configuration Set Migration page, select Configuration > Migration from the Navigator menu.

This report includes all new or updated:

- Objects
- Fields
- Pages
- Business intelligence (BI) changes

Business logic changes such as Groovy scripts and triggers aren’t included in the configuration set report.

### Related Topics

- Implementation Project Based Export and Import: Explained
- Moving Related Common Reference Objects: Points to Consider
- Context Layers: Explained

### Migrating Global Search Configurations: Points to Consider

The configuration set, which you create using the Configuration Set Migration page, doesn't include global search configurations. After importing a configuration set, if global search is configured in the source environment, you must manually reset the configurations in the target environment.
Manually Resetting Global Search Configurations
You must adhere to the following key guidelines for manually resetting global search configurations:

- If a search view object was activated in the source environment, then after migration, you must activate the object in the target environment. Use the Manage Search View Objects task to activate the object.

- If a search view object was deactivated in the source environment, then after migration, you must deactivate the object in the target environment. Use the Manage Search View Objects task to deactivate the object.

- The process of activating and deactivating a search view object affects the categories that are shown in global search. Hence, after activating or deactivating any search view object (after migration), you must adjust the categories to reflect the state of the source environment. Use the Manage User Interface task to adjust the categories to be shown in global search.

- If the category name for a search view object changes, then after migration, you must update the category name. Use the Manage Search View Objects task to edit the search view object and thereby, update the category name.

- If the schedule for a search view object changes, then after migration, you must update the schedule. Use the Manage Search View Objects task to edit the search view object and thereby, update the schedule.

- If any search view objects were reconfigured in the source environment, then the source environment automatically triggers a reindex. After migration, you need not reconfigure the search view objects. However, you must trigger a full reindex in the target environment for any search view objects that were reconfigured in the source environment. To trigger full reindex, select the search view object in the Manage Search View Objects task, and click the Full Re-Index button.

Exporting and Importing Supported Configurations: Explained
After you have finished creating application changes in Application Composer, you can export and import them across Oracle application instances on the same release. Use Configuration Set Migration to export and import application changes.

Some of the supported application changes that you can export and import include object UI extensions, object server scripts, saved searches, workflows, global functions and so on. Don’t create these supported changes manually in the target application instance. Import these supported changes from the source instance only.

⚠️ Note: In the target application instance, you must create only unsupported application changes.

To view the application changes in the target application instance, you must first export your changes from the source instance as compressed files in .jar format. You can then import these compressed files in the target instance.

The following sections provide an overview of exporting and importing application changes. For more information on using Configuration Set Migration, see "Using Configuration Set Migration to Move Configurations: Points to Consider."

⚠️ Note: In Configuration Set Migration, the changes are directly copied to the mainline metadata and no sandbox is involved.

Exporting Configurations
You can export application changes by creating a configuration set using Configuration Set Migration as compressed files in .jar format.

When exporting, keep in mind the following points:

- Don’t modify the compressed files.
• Don’t move application changes from the target application instance back to the source application instance.

To export application changes:

1. In the target application instance, click **Navigator > Configuration > Migration.**
   
   The Configuration Set Migration page opens.

2. On the Outgoing tab of the Configuration Set Migration page, click **Create Configuration Set.**

3. Provide a name for the configuration set.

4. Optionally, type a description of the set.

5. Select the content you want to migrate.

   **Note:** The Industry solution extensions module is for Oracle’s internal use only and has no impact on Configuration Set Migration operations.

6. Click **Save and Close.**

7. Click **Refresh** periodically to see the current status of the set creation because creating a configuration set can take a few minutes.

   You can click **Log** to review the process log, which provides more details about the configurations that are being compressed. If the process terminated with a status of Error, the log would provide you information about the configurations that failed to compress. The process runs asynchronously, so you can close the page and return to it later. Eventually, the status changes to Ready for Download, which means that the configurations are ready for you to download. Before downloading the configurations, you can click **Content Read Me** to download the Readme file listing all the configurations you exported.

8. Click **Download,** and specify the name and location for the file, and click **Save.** Ensure that the downloaded file is a .jar file.

9. After you download the file on your local file system, click **Delete** to remove the temporary files that were created on the server.

**Importing Configurations**

After you have successfully exported the application changes to a compressed file format, you can import the compressed file into the target application instance.

To import application changes:

1. In the target application instance, click **Navigator > Configuration > Migration.**

   The Configuration Set Migration page opens.

2. Click the **Incoming** tab.

3. Click **Browse** to select the configuration set archive file.

4. Click **Apply.**

5. Periodically, click **Refresh** to view the current status of the Apply action.

   You can review the process log, if required. The process runs asynchronously, so you can close the page and return to it later. When the configurations are applied, the status is displayed as Applied and Deleted.

   If problems occur during an Apply action, log a service request using My Oracle Support at https://support.oracle.com.

If you encounter any errors during the import process, (for example, the connection to the database is lost or process issues occur during migrating security policies), then all changes roll back automatically, reverting the target application instance to the version before the import.
If you have made any security changes in the source instance outside of Application Composer, ensure that you manually re-enter these security changes in the target instance prior to using Configuration Set Migration for importing. For example, if you have set up a custom security role in the source instance, ensure that you manually add this security role in the target instance, prior to using Configuration Set Migration.

Note that importing earlier versions of compressed files does not roll back changes in the target instance to an earlier version.

When importing, if you upload a file in a format other than .jar, then no warning message appears but the import job is processed and fails with an error status.

**Migrating FND Lookups**

You can use Configuration Set Migration to migrate FND lookups.

**Supported Application Composer Changes and Best Practices: Explained**

You can export and import supported application changes across Oracle Sales Cloud applications that are on the same release and same patch level. Use Configuration Set Migration to export and import supported application changes. To access Configuration Set Migration, select Migration from the Configuration section in the Navigator menu in the source or target environment.

This topic describes the following:

- Supported application changes
- Unsupported application changes
- Best practices for using Configuration Set Migration

**Supported Configurations**

The supported application changes for Application Composer include:

- Object UI extensions
- Object server scripts
- Saved searches
- Workflows
- Global functions
- Object model extensions
- Relationships
- Role security privileges to access objects
- E-mail templates
- Custom subject areas: Republish all custom subject areas in the target Oracle Sales Cloud application. See "Publishing Custom Subject Areas: Explained."
- FND lookups
- All reports, analyses, and dashboards
- Migrate using FSM tasks: Manage Standard Lookups, Manage Custom Lookups, and Manage Set-Enabled Lookups.
Unsupported Configurations

Application changes that are not supported for Application Composer include:

- Import and export artifacts generated for custom objects and fields: Regenerate manually using Application Composer’s Import and Export menu option.

Application changes made outside Application Composer and not supported include:

- Sales Prediction Engine business rules: Recreate manually in the target Oracle Sales Cloud application.
- Security job roles and duty roles: Recreate manually in the target Oracle Sales Cloud application.

Note: Application Composer supports Page Composer modifications.

You must create unsupported application changes manually in the target application.

Best Practices for Using Configuration Set Migration

Best practices for using Configuration Set Migration include:

- Do not manually create supported application changes in the target.
- Manually create application changes that are partially supported or not supported by Configuration Set Migration.
- Use Configuration Set Migration to migrate FND Lookups.
- Do not modify metadata extract.
- Migrate metadata from one instance to another.
- Import the most recent extract.
- Manually re-key security changes in the target environment using Security Console prior to using Configuration Set Migration.
- During an export or import, you must not make changes in the source or target instance.

Viewing and Deleting Configurations: Procedure

Use the Manage Customizations dialog box to view the changes made to the application pages and to delete unwanted changes. Click your user image or name in the global header, and select Manage Customizations from the Administration menu.

Deleting Configurations

To delete configurations:

1. On the page that contains the configurations, select the page fragment or task flow, and then select Manage Customizations from the Administration menu.
2. In the Name list, select the correct layer, and find the page, task flow, or fragment that contains the configurations.
3. Click Delete for the configuration document that you want to delete.
4  Adding Objects and Fields in Application Composer

Using Application Composer: Overview

Use Application Composer to create fields, objects, and relationships. Then, modify user interface pages for all users, or only some. Common tasks available in Application Composer are described in this chapter.

This chapter provides a summary of how Application Composer works and explains:

- How to define custom objects
- How to define custom fields for either a custom object, or a standard object
- How to secure both the actions in an object’s work area, as well as the data that users can see

Other chapters in this guide describe additional tasks flows that are available in Application Composer. For example, learn how to create object workflows and custom subject areas, how to write Groovy scripts, and how to import and export your application changes. Refer to the table of contents for these other chapters.

Also refer to the Oracle Sales Cloud Groovy Scripting Reference guide, available in the Oracle Sales Cloud documentation library, for specific examples and use cases about changing your implementation of Oracle Sales Cloud.

You can access Application Composer by selecting Application Composer from the Navigator menu, under the Configuration category. To test your changes, use the Navigator to switch to the desired application.

Tip:

Navigate quickly and easily between Oracle Sales Cloud run time pages and Application Composer design time pages using the Favorites and Recent Items menu.

Related Topics

- Defining Pages: Explained

Extending Oracle Sales Cloud: How It Works

Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application developers, to extend Oracle Sales Cloud. Make the type of data model changes which, for non-Sales Cloud applications, can only be made by application developers. For example, easily create a new object and related fields, then create new user interface pages where that object and its fields are exposed to users.

Application Composer is a design time at runtime tool, which means that you can navigate to Application Composer directly from any Sales Cloud application, make your changes, and see most changes take immediate effect in real time, without having to sign out and sign back in.
Note: To see your changes in real time, always use the Navigator to navigate to the run time page that you changed. Then navigate back to Application Composer to continue making changes. In other words, when making application changes (and testing them), restrict your usage to a single tab. Don’t work across multiple browser tabs, because Application Composer doesn’t support this type of usage.

Application Composer is supported for use only in English. Additionally, Application Composer is not supported for use with iPad devices.

Application Changes for Nondevelopers

Application Composer hides the complexity of modifying applications by leveraging a set of standard design patterns and wizards. You focus on the application changes that your business requires (object model extensions and layout changes, for example), and Application Composer creates the underlying object artifacts for you.

Using Application Composer, you can make application changes such as:

- Modifying objects by adding new fields, or create entirely new objects.
- Creating foreign key-based relationships between two objects.
- Modifying user interface pages by exposing your newly created fields for an object, or create an entirely new work area for your custom objects.
  
  Expose object relationships on desktop pages in the form of subtabs.
- Writing application logic, such as triggers, validation rules, and workflows, for an object or for use across multiple objects.
- Implementing functional and data-level security for custom objects.
- Enabling objects for custom reporting.
Accessing Application Composer

Access Application Composer from any Sales Cloud application at run time by using the Navigator menu, and selecting **Application Composer** under the Configuration category. The first view of Application Composer is the main Overview page, which is the entry point into all your task options.

**Getting Started**

From the main Overview page:

- Use the object tree to select the object you want to modify. Or, click the New icon to create a new object.
• Use the links in main Overview page, also known as the local area, to select a task. Or, use the links in the Common Setup pane.

Related Topics
• Defining Pages: Explained

Viewing Application Composer Changes: Explained

Use the Configuration report to view a detailed list of Application Composer changes made by administrators in your environment. Access the report under Metadata Manager in the Common Setup pane in Application Composer. You can generate the report as many times as needed, each time overwriting the previously generated version of the report.

Configuration Report

The Configuration report lists changes in your environment that were created in Application Composer, as described in the following list. You can download the report in HTML, in Microsoft Excel (.xls) format, or in Microsoft Excel Worksheet (.xlsx) format.

The report includes:

<table>
<thead>
<tr>
<th>Report Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Summary</td>
<td>Provides a summary of changes such as the number of objects and fields, relationships created, and object validations.</td>
</tr>
<tr>
<td>Global functions</td>
<td>Includes a listing of the global functions created.</td>
</tr>
<tr>
<td>Relationships</td>
<td>Includes a listing of all the relationships created.</td>
</tr>
<tr>
<td>Object summary</td>
<td>Select All modified objects to include all the changes that have been made to any objects.</td>
</tr>
</tbody>
</table>
Adding Objects and Fields in Application Composer

<table>
<thead>
<tr>
<th>Report Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select</strong> Selected objects</td>
<td>Select to include all the changes for only a subset of objects.</td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>To include layout details, select up to five objects only.</td>
</tr>
<tr>
<td>Fields</td>
<td>Includes custom fields and standard fields that have been modified for the objects that you selected.</td>
</tr>
<tr>
<td>Server scripts</td>
<td>Includes object validations, object functions, object triggers, and global functions for the objects that you selected.</td>
</tr>
<tr>
<td>Object workflows</td>
<td>Includes object workflows for the objects that you selected.</td>
</tr>
<tr>
<td>Page layout summary</td>
<td>Includes a listing of dynamic page layout names and layout conditions for the objects that you selected.</td>
</tr>
<tr>
<td></td>
<td>Select <strong>Layout details</strong> to include a list of all the actions, buttons, subtabs, and field names exposed in each subtab of a dynamic page layout.</td>
</tr>
</tbody>
</table>

To submit the report:

1. Confirm that you are not in a sandbox session.
2. In Application Composer, select **Metadata Manager** in the Common Setup pane.
3. Click **Generate**.
4. In the Generate Configuration Report dialog, indicate the items that you want to include in the Configuration report.
5. Click **Generate**.
6. Download the report by selecting either the HTML or Excel format.

Working with Objects

Defining Objects: Explained

Using Application Composer, modify an Oracle Sales Cloud application's object model so that you can track and store any additional data you might need. For example, add new fields to an existing object (standard objects), or create entirely new objects (custom objects). Standard objects are objects that are delivered with a Sales Cloud application, and made available to Application Composer for application changes. Custom objects are objects that you create using Application Composer. You can create either top-level objects (objects without a parent) or child objects (objects created in the context of a parent).

Read this topic to learn about these tasks:

- Browsing the object tree
- Creating a custom object
- Using the Object Overview page
- Editing an object's attributes
- Selecting the display icon for the object's set of UI pages
- Viewing child and related objects
• Deleting a custom object

Using Application Composer’s Object Tree

Access Application Composer from the Navigator menu. The first view of Application Composer is the main Overview page, which is the entry point into all your application change options.

On the main Overview page, the regional pane at left displays the object tree, which lets you browse an application’s existing object configuration in a tree format. The object tree reflects the latest configuration of the application: both standard objects as well as custom objects.

To use the object tree:

1. Select Application Composer from the Navigator menu, under the Configuration category.
2. For each object node, whether standard or custom, expand it further to view and edit object details.

For example, look at object details such as fields and UI pages where the object is exposed.

Note: At the top of the object tree, you can also click the New icon to create a new custom object.
For both standard and custom objects, you can view and edit the following details:

- **Fields**
  Add new fields to an object.

- **Pages**
  Modify the pages on which an object appears.

- **Actions and links**
  Add actions or links to pages.

- **Server scripts**
  Write application logic that controls the behavior of an object's records.

For custom objects, you can also view and edit details to security. For example, you can implement functional and data-level security for an object and its records.

### Creating a Custom Object

Create a custom object if you want to track data about an object that’s not already delivered with Oracle Sales Cloud. After you create the object, you then add custom fields and design user interface pages where your users can enter object records. There is no fixed limit on the number of custom objects that you can create.

To create a custom object:

1. On the main Overview page of Application Composer, select the **Custom Objects** node in the object tree, or click the icon in the local area of the main Overview page.
2. On the resulting summary table, click the New icon, or at the top of the object tree, click the New icon.
3. Complete the primary identifying attributes for a custom object:

   - **Display Label**
     An object's display label is the user-friendly label for an object, and also becomes the default page title for the object's work area.

   - **Plural Label**
     The plural label is used as the title of the object's work area. The label is also used as the search string in the regional search, as well as in the saved search on the object's run time overview page.

   - **Record Name Label**
     Use the **Record Name Label** field to specify the display label for the object's RecordName attribute. The RecordName attribute stores the user-entered "name" of the record. For example, if you're creating a custom object, Book. In the **Record Name Label** field for this object, you would enter something like "ISBN Number." At run time, for each new record, your users would use the **ISBN Number** field to enter the book's International Standard Book Number (ISBN), which uniquely identifies books published internationally.

     Typically, this field is the object's primary user-recognizable identifier that run time users drill down on, from the landing page to the detail page. For example, at run time, your users would click any ISBN to drill down to review details about the book, such as book title and author.

   - **Record Name Data Type**
     Select either Text or Automatically Generated Sequence.

For custom objects, you can also view and edit details to security. For example, you can implement functional and data-level security for an object and its records.
For record names of Text data type, the maximum length that users can enter is 32 characters. For record names of Automatically Generated Sequence data type, the sequence number is based on a display format which is up to 28 characters and has at least one number token: {0}.

**e.** Select the **Prevent duplicate values?** check box to prevent users from entering records with duplicate names.

   **i.** If the **Prevent duplicate values?** check box is selected, then the **Treat "ABC" and "Abc" as distinct values** check box is enabled.

   Select this check box if you want the assessment for duplicate records to be case sensitive.

**f. Object Name**

The object name is the internal name for the object.

![Note:](image) You can use a custom object’s internal name only once across the mainline code and all existing sandboxes. If you previously used an object’s internal name in a sandbox, you can reuse that same internal name, but you must first you delete all other sandboxes where the internal name was previously used. You can use a custom object’s display name as many times as you want across sandboxes. The restriction applies only to the internal name.

**g. Description**

4. **Click OK.**

Once your custom object is created, you must add fields and then create the UI pages where your users can create actual records. See Defining Fields: Explained" and "Creating a Set of Simplified Pages for Custom Objects: Explained.

**Tip:** To create a custom child object, click the Create Child Object button from the parent object’s Object Overview page. See the next section in this topic. Once created, a child object cannot be changed to a parent object. Similarly, a parent object cannot be changed to a child object. Child objects are discussed below.
Using the Object Overview Page

The Object Overview page provides a high-level overview of a standard or custom object. The Object Overview page displays the primary attributes for an object, plus a list of child objects and related objects, if any.

To access the Object Overview page:

1. On the main Overview page, select the object in the object tree or select the Standard Objects or Custom Objects node in the object tree, or the icon in the local area of the main Overview page.
2. Select the object from the resulting summary table, and click the Edit icon.

From the Object Overview page, you can:

- Edit the object’s primary attributes, described in the previous section. For example, change the Display Label or Record Name Label.
- Change the display icon for the object.
  
  This process is described as follows:
- View the parent child relationships that were created for this object.
  
  You can also create new child objects from this page, which implicitly creates a new parent child relationship.
- View the non-parent child relationships that were created for this object.

Editing an Object’s Attributes

After an object has been created, you can edit its attributes from its Object Overview page.

To edit an object’s attributes:

1. Access the Object Overview page for the object, as described earlier.
2. On the Object Overview page, click **Edit**:
   - Change the object’s primary attributes, such as display label, description, and record name, at any time.
   - You cannot change the Object Name and API Name after the object has been created.

A custom object’s API name is automatically derived using the logical name followed by `_c`. You use the API name in Groovy expressions that you build with the expression builder, when writing business logic for the object.

### Selecting the Display Icon for Objects

From the Object Overview page, you can select the display icon to use for the object’s UI pages. You can select the display icon for custom objects (although a default icon is provided), and you can change the icon for standard objects. The icon you select determines which icon and theme display to your end users in a variety of locations, such as on the Navigator, subtabs, mobile pages, and the springboard strip on simplified pages.

**Tip:** The icon selected for standard objects is inherited throughout Oracle Sales Cloud. For example, if you change the display icon for the Opportunity object, then all UI pages are automatically updated to the new icon. This includes even custom subtabs that you added using Application Composer.

To select the icon:

1. Click the object’s node in the object tree to view the object’s Overview page.
2. On the Overview page, set the icon for the object in the Display Icon region.
Viewing Child and Related Objects

The Object Overview page displays a list of child objects and related objects, if any, that have been created for an object. You can also create new child objects from this page.

- A child object is an object with a cascade delete relationship to a parent object. This means that if you delete the parent object record, then all its child records are automatically deleted. A child object does not exist outside the context of the parent object, and does not have its own work area. You cannot change a child object to a parent object after the child object has been created.

- Related objects can exist independently of each other, even if one object is deleted. Related objects are connected in a foreign key-based relationship between two top-level objects, not as parent and child. These types of relationships include reference relationships and dynamic choice list relationships.

Related objects can have either a one-to-many or a many-to-one relationship with the current object. Note that an object can be related to itself to model a hierarchy of the object. In this case, the object itself is displayed on its Object Overview page as a related object. For example, the Department and Sub-department objects would be displayed in this way.

Note: You do not create these types of relationships from this page. Instead, manage relationships from the Relationships page, which you can access from Application Composer’s main Overview page. Or, create a dynamic choice list relationship by creating a dynamic choice list field for an object, which derives its choice list values from another object.

To create a child object for a standard or custom object:

1. Navigate to an object’s Object Overview page.
2. Click the **Create Child Object** button. Creating a child object is the same as creating a custom object, except:
   - The current object is automatically displayed as the parent object.
   - Specify the **Child Collection Name** field to specify the internal name for this set of child object records, which can be used later when writing Groovy scripts.

Deleting a Custom Object

Application Composer does not support the deletion of either standard or custom objects. If you no longer need an object that was already published to the mainline metadata, optionally enter a note in the description that the object is no longer used.

Related Topics

- Creating a Work Area: Explained
- Defining Pages: Explained
- Creating a Set of Simplified Pages for Custom Objects: Explained

Object Relationships: Explained

Your end users will often need to associate one object’s records with the records of another object. To enable this type of association between records, you must first create a relationship between those two objects. For example, maybe your users want to track the opportunities that get created for an account.
In this example, you will create a one-to-many relationship between the account and opportunity objects, and then expose a list of opportunities as a subtab on the account’s details page. This lets users search for and add one or more opportunities to a single account record. When creating relationships between objects, there are four types of relationships that you can pick from in Application Composer. Each type of relationship has its characteristics and advantages. In general, they all let you use a subtab to create or assign one or more records from one object to a record from another object.

Review these aspects of using and managing relationships in Application Composer before you begin to create relationships between objects:

- Relationship types
- Adding subtabs

**Relationship Types**

Application Composer lets you create either a one-to-many relationship, or a many-to-many relationship. Across these two categories, there are four types of relationships that you can pick from when creating a relationship.

- **Parent child relationship.** A parent child relationship is a one-to-many relationship: one parent record can have many children records. When you create a child object, it’s created specifically in the context of its parents. A child object doesn’t have its own work area, and the child object’s records are deleted if the parent objects is deleted.

- **Dynamic choice list relationship.** A dynamic choice list field provides a list of values from a source object, which your users can select and associate with a target object. When you define the dynamic choice list field, Application Composer automatically creates a one-to-many relationship between the source object and target object. This means that not only do you get the ability to associate a source object and target object records that are associated with a single source object record.

- **Reference relationship.** You can also manually create a one-to-many relationship, where you can specify a source object and target object. Thus, this type of relationship is similar to a dynamic choice list relationship. The only drawback is that you don’t get a dynamic choice list field to add on the target object’s work area. A reference relationship only gives you the ability to add a subtab to the source object’s details page, showing a list of all the target object records that are associated with a single source object record.

- **Many-to-many relationship.** Create a many-to-many relationship where, similar to a one-to-many relationship, you can specify a source object and target object. However, with one-to-many relationships, you can add a subtab only to the source object’s details page. This lets your users add one or more records from one object to one or more records from another object.

**Adding Subtabs**

After you create relationships between objects, you can then expose one object’s records on a subtab that is displayed on the other object’s details page.

When adding a subtab to an object’s details page, you select to add a Child or Related Objects subtab from the object’s Pages Overview page. Application Composer lets you add a subtab based on any target object that has a relationship with the current object as the source object. Subtabs are discussed in related topics.

**Related Topics**

- Subtabs: Explained
- Tree Nodes: Explained
One-to-Many Relationships: Explained

Using Application Composer, you can create one-to-many relationships between two objects within the same application, where one object’s primary identifier is stored in another object’s table. A relationship must exist before you can expose the “many” objects on a subtab that is displayed on the “one” object’s details page or tree. For example, an account can have multiple service requests associated to it. To expose a list of service requests associated with a specific account as a subtab on the account’s details page, you must first create a one-to-many relationship between the account and service request objects. Create these relationships implicitly by creating a child object or by creating a dynamic choice list. Or, create relationships explicitly on the Create Relationship page.

Parent Child Relationships

Parent child relationships are implicitly created when a custom object is created as a child of a top-level object.

When a child object is created, it is created specifically in the context of its parent. A child object does not have its own work area, and the child object is deleted if the parent object is deleted.

View parent child relationships in the object tree, where child objects appear as sub-nodes beneath their parent objects. If a parent child relationship exists, then the child object is listed on the parent’s Object Overview page in the Child Objects region. A top-level object can have many child objects. A child object can have only one parent object.

Relationships that are implicitly created from parent child relationships are also displayed on the Relationships page. The relationship name is automatically generated for you.

Dynamic Choice List Relationships

Choice list relationships are implicitly created between two objects when you create a dynamic choice list field.

A dynamic choice list is a field that contains a list of values which are populated from the actual data of another object. For example, you might want to expose on a desktop page a dynamic choice list which lets users specify the HR representative of a given department. The HR Representative choice list is a field that you are adding to the department object, but the list of values is populated by actual employees from the employee object.

When you select an object and create a dynamic choice list field based on a related object, you are implicitly creating a one-to-many foreign key relationship where the current object is the “many” object and the related object is the “one” object. This implicit creation of a relationship lets you later add a related object subtab for the “many” object on the “one” object’s details page. You can view these implicitly created choice list relationships on the Relationships page.

View dynamic choice list relationships on an object’s Object Overview page. If such a relationship exists, then the related object is listed on the object’s Object Overview page in the Related Objects region.

These objects are related objects, not parent child objects; related objects are not deleted if the current object is deleted.

Relationships that are implicitly created from dynamic choice list relationships are also displayed on the Relationships page. The relationship name is automatically generated for you.

Note: Generally, the dynamic choice list that you create results in the implicit creation of a choice list relationship. The exception is if you are in a global single instance environment and you create a dynamic choice list between a Sales Cloud object and a common object: resource, customer contact profile, account, address. In such cases, relationships are not implicitly created.
Creating Reference Relationships

Create a foreign key-based, one-to-many relationship between two top-level objects explicitly using the Create Relationship page. This type of relationship is called a reference relationship.

To explicitly create a relationship between two top-level objects within the same application:

1. Select Relationships in the Common Setup pane.
2. On the Relationships page, click the New icon.
3. Select the source object and target object.

A child object cannot be the source object or target object.

The Note common component is not available for selection as either a source object or target object.

Once you create a relationship, you can no longer edit the source and target objects.

This relationship adds a field to the target object to store the foreign key details. If the source object is ever deleted, the target object records remain.

4. Enter the relationship name and description.

Once you create a relationship, you can no longer edit the relationship name.

5. Optionally add the target object in a subtab to the source object’s detail page.

Note: You can create multiple relationships between the same source and target objects. For example, create both a Primary Contact and Secondary Contact relationship between the contact and opportunity objects.

Groovy Script Syntax

Once you have created a one-to-many relationship between objects, a foreign key field is created on the child object or on the “many” object. Use the following API names to access those foreign keys in your scripts.

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Foreign Key API Name</th>
<th>Pattern Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent/child relationship</td>
<td>If the parent object name is ParentObj_c, then the foreign key API name (added to the child object) is ParentObj_c.Id_c.</td>
<td>&lt;Name of the parent object&gt;_Id_c</td>
</tr>
<tr>
<td>Dynamic choice list relationship</td>
<td>If the dynamic choice list field name is DynChoice1_c, then the foreign key API name is DynChoice1_c.Id_c.</td>
<td>&lt;Name of the dynamic choice list field&gt;_Id_c</td>
</tr>
<tr>
<td>Reference relationship (one-to-many)</td>
<td>If the source object name is SourceObj_c, the target object name is TargetObj_c, the relationship name is relation_Mto1, then the foreign key API name (added to the target object) is SourceObj_c.Id_c.relation_Mto1.</td>
<td>&lt;Name of the source object&gt;<em>Id</em>&lt;Name of the relationship&gt;</td>
</tr>
</tbody>
</table>

Related Topics

- Subtabs: Explained
Many-to-Many Relationships: Explained

In addition to one-to-many relationships between objects, objects can also have a many-to-many relationship between each other. For example, a service request can have multiple employees working on it. At the same time, a single employee can work on multiple service requests. In this scenario, you would create a many-to-many relationship between the Service Request and Resource objects, where the related records from both objects store their primary identifiers in an intersection object. Many-to-many relationships are not supported in desktop work areas.

Creating Many-to-Many Relationships: Example

To create a many-to-many relationship using Application Composer:

1. Select Relationships in the Common Setup pane.
2. On the Relationships page, click the New icon.
3. Select the source object and target object.
   
   A child object cannot be the source object or target object.

   The Note common component is not available for selection as either a source object or target object.

   Once you create a relationship, you can no longer edit the source and target objects.

   ✍️ Note: You can create only one many-to-many relationship for a particular set of objects.

4. Enter the relationship name and description.
   
   Once you create a relationship, you can no longer edit the relationship name.

5. Indicate the cardinality of the relationship:
   
   • M:M
   
   Many-to-many

6. Enter the name of the intersection object.
   
   The intersection object’s table records two foreign keys: one for the Service Request object and the other for the Resource object. This enables the many-to-many relationship.

   The intersection object is available as an extensible, top-level object in Application Composer. Optionally extend the intersection object. Custom fields that you add to the intersection object are available for display on the subtabs you create, which is discussed in the section below.

   Intersection objects also have a corresponding web service automatically published.

7. Optionally specify data filter criteria for both the source and target objects.
   
   The filter criteria that you specify here controls which records are available for association at run time with a record from the other object in this relationship.

   Read: "Configuring a Search and Select Dialog for Custom Objects".
Adding Subtabs

After you create the many-to-many relationship, you can now add related object subtabs on each object’s simplified details page:

> **Note:** You can add subtabs for a many-to-many relationship to simplified details pages only. Many-to-many relationships are not supported in desktop work areas.

- Create an Employee subtab on the service request’s details page.
  The subtab displays all employees that are working on a specific service request. At run time, your end users can add or remove employees who are working on a specific service request.
  When creating the subtab, you can select which Resource fields to display, such as Employee Name and Title. You can also select which intersection object fields to display, such as Primary Service Request Owner.

- Create a Service Requests subtab on an employee’s details page.
  The subtab displays all service requests that an employee is working on, since each employee can work on multiple service requests. At run time, your end users can add or remove service requests that an employee is working on.
  When creating the subtab, you can select which Service Request fields to display, such as Service Request Abstract and Date Logged. You can also select which intersection object fields to display, such as Primary Service Request Owner.

When selecting the fields for display on a related object subtab, join fields are not available for selection if the relationship is a many-to-many relationship.

Configuring a Search and Select Dialog for Custom Objects

A Search and Select dialog, also known as a picker, lets your end users search for and select object records when assigning one record to another, such as an employee to a service request. These dialog boxes are launched from the related object subtabs that you create, after creating relationships.

Search and Select dialog boxes are automatically provided for standard objects, and are not extensible. However, if you’re creating a many-to-many relationship that involves a custom object, then you must configure the Search and Select dialog boxes for those custom objects.

The filter criteria that you specify in the relationship definition applies to the Search and Select dialog, and controls which records are available for association at run time with a record from the other object in this relationship.

For example, you can define filter criteria that lets your end users select only "unassigned" service requests for association with an employee.

**Groovy Script Syntax**

Once you have created a many-to-many relationship, two foreign key fields (one for each object) are created on the intersection object. Use the following API names to access those foreign keys in your scripts.

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Foreign Key API Name</th>
<th>Pattern Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference relationship (many-to-many)</td>
<td>If the source object name is SourceObj_c, the target object name is TargetObj_c, and the intersection object is IntersectionObject_c then the two foreign key API names (added to the intersection object) are TargetObj_Id_Tgt_, TargetObj_cToIntersectionObject_</td>
<td>&lt;Name of the object&gt;<em>Id_Tgt</em>&lt;Name of the object&gt;<em>cTo&lt;Intersection object name&gt;.</em>&lt;Name of the object&gt;<em>Id_Src</em>&lt;Name of the object&gt;<em>cTo&lt;Intersection object name&gt;.</em></td>
</tr>
</tbody>
</table>
Activating Global Search on Objects You Created or Deactivated: Procedures

Oracle activates global search on all application objects where search is available. Use this procedure to activate search on any objects you deactivated in the past or for objects you created. You can activate search only on the objects you created, not on their child objects.

To make an application object available for global search, you must do the following:

1. Activate the object.
2. Specify the frequency with which the object will be indexed.
3. Optionally, you can modify the list and order of fields indexed in the search and displayed in the search results.

Activating an Object for Search

To activate an object for search, do the following:

1. While signed in as a setup user or a sales administrator, navigate to the Setup and Maintenance work area.
2. In the Setup: Sales page **Functional Areas** column, select the **Sales Foundation**.
3. In the **Sales Foundation** column on the right, click the **Manage Search View Objects** link.

   The Manage Search View Objects page appears.
4. Select the object you want to enable for search.
5. Click **Activate**.

   The status for the object changes to **Active**.

**Tip:** Make sure you deactivate any object that is not needed for global search to maximize system resources.

Setting the Indexing Frequency and Schedule

After you have activated the object, you must specify how frequently you want the object records indexed.

Oracle recommends that you index objects daily during off-hours. You should stagger the indexing times for the different objects to minimize performance impacts.

Specifying the fields to be indexed and displayed in the search results is optional because these are already set up for you.

1. Select the **Display Name** link of the object.

   The Edit Search View Object page appears.
2. In the Index Schedule region, select the **Frequency Type** and enter the number of days between index runs and the time, if appropriate. Oracle recommends staggering the indexing schedule to maximize available system resources.

3. You can change which fields the application indexes and which fields display in search results as described in the Specifying Which Fields Are Indexed and Displayed in Search Results section.

4. When you are done, click **Save and Close**

   The application returns you to the Manage Search View Objects page where you can monitor the status of the index generation for each object.

   The first time your scheduled indexing process runs or any time you modify the list of fields in the object, the application generates a complete index of all the existing records. Subsequently, the process indexes only records that have changed.

   If you end up with many inactive records in your system over time, you can improve the efficiency of your searches by periodically regenerating the full index. This can be accomplished by selecting the object and clicking **Full Reindex**.

Specifying Which Fields Are Indexed and Displayed in Search Results

In the Edit Search View Object page, you can also change which fields the application indexes and which fields display in search results. You must index the fields you created if you want them to be available for searches.

- **Title** and **Fixed Content** fields let you specify which fields are displayed in search results and in what order.
  - **Title** is the linked heading of each search result.
  - **Fixed Content** is the text which appears under the heading.

  For example, the titles starting with the word **Opportunity** are links which permit users to drill down to the record. The rest of the fields are the fixed content.

- **Body** field lists the fields that are indexed by the application. The most relevant fields are displayed in the search results, space permitting. While the **Body** field includes all of the standard fields for indexing, you must add the fields you created to the list if you want them available for searches.

To make changes, click **Edit** (the pencil icon) and make your changes in the Edit Search View Object window.

**Related Topics**

- Enabling the Global Search Profile Option
- Changing the Behavior of Global Search Automatic Suggestions

Managing Security for Custom Objects

Making Custom Object Pages Visible to Users: Explained

Use Application Composer to create custom objects and fields, as well as the user interface (UI) pages where your users can enter data. By default, a custom object and its records are visible and editable only to users who are provisioned with the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. You must have this custom objects role assigned to you, before you can view and test custom objects in the sandbox. After creating custom objects, you must indicate which end users can view the pages and enter data. Grant additional access manually in Application Composer using the custom object’s Security node, or the Role Security link in the Common Setup pane.
Review these aspects of the custom object security process in Application Composer before you begin to define your security policies:

- Who can create custom objects?
- Who can see your custom object?
- Understanding security policies
- Custom vs. predefined roles
- Application Composer and the Security Console

### Who Can Create Custom Objects?

Users with any one of the three following job roles can create custom objects and use all other Application Composer functions:

- Customer Relationship Management Application Administrator.
- Application Implementation Consultant.
- Master Data Management Application Administrator.

Oracle recommends provisioning the user with the Customer Relationship Management Application Administrator job role (for performing the application changes) and the Custom Objects Administration job role and Sales Administrator job role (for testing the application changes in the Oracle Sales Cloud UI).

### Who Can See Your Custom Object?

When you create custom objects, by default their UIs are visible only if you have the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. The application creates this custom role automatically. The UI pages you create for custom objects are not visible to additional users unless you provide access in Application Composer using the object’s Security node. Use the Security node to specify not only which job roles can access the UIs, but the levels of access. For example, you can make it possible for sales managers to update object records, while sales representatives can only view records.

To manage who can see your custom objects:

1. Ensure that you have assigned the Custom Objects Administration role to yourself and to other users who make application changes.

   The Custom Objects Administration role is automatically assigned to new custom objects, as well as existing custom objects, if you have upgraded from a previous release.

   See "Enabling the Testing of Custom Objects in the Sandbox: Procedure."

2. For each custom object, use the Security node to specify which roles can view that object’s UI pages, and their level of access: view, update, and delete. This is called a security policy. See the following "What’s a Security Policy" section.

   When granting access to custom object UIs, you can select only custom job roles. For example, if you want to create a custom object for sales managers, then a custom sales manager job role must first exist (instead of the predefined Sales Manager job role provided by Oracle), before you can grant access to sales managers. If you need to create a custom job role, then copy the predefined Sales Manager job role in the Security Console as described in the Oracle Sales Cloud Securing Sales guide.

   Granting access to custom job roles means that your custom object won’t be affected by future upgrades. See the "Custom vs. Predefined Roles" section later in this topic.

3. If you’re creating custom objects for a specific job role, then you must also assign yourself that job role to view and test the application changes in a sandbox. For example, if you’re creating a custom object for sales managers, then you must assign yourself the sales manager job role to test how that object works for sales managers. If you later
create a different object for sales representatives, then you will have to deprovision the sales manager job role and provision yourself with the sales representative job role instead, so that you can accurately test your new object.

- Setup users, who have the permission to create and update users, can grant themselves the appropriate job roles by editing their user record in the Manage Users work area.
- Sales administrators, who are resources, can request the job role they need for testing by following the procedure described in "Assigning Yourself an Additional Job Role."

To make job roles requestable for sales administrators, a setup user must create a special role-provisioning rule, as described in "Creating the Provisioning Rule for the Job Roles Used in Testing."

4. If you’re adding a custom object subtab to a standard object, then you must also assign yourself the job role that can view the standard object’s UI.

For example, let’s say you add a custom object subtab to the Edit Opportunity details page. In this case, you will need the role required to access the Edit Opportunity page, in addition to the role granted to the custom object.

**What’s a Security Policy?**

For each custom object, you will need to update its security policy. A security policy specifies which users are authorized to access an object’s data, and what type of access they have. Access includes both function security as well as data security. For example, does a user have view only access, or can the user create and update an object’s record, as well?

As previously mentioned, custom objects are automatically assigned the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. Next, grant additional access to your custom object so that your end users can enter data.

For each custom object, you can grant access to multiple roles for a single object, or you can grant access to multiple objects for a single role.

- Define security policies **for an object.**
  Authorize the various custom roles whose users can access that object’s data.
  You must define security policies for child objects, as well.
  See "Managing Security by Object: Explained."

- Or, define security policies **for a role.**
  Specify the role’s access levels across multiple custom objects.
  See "Managing Security by Role: Explained."

Define the security policy for a custom object using the Security node in Application Composer, on the Define Policies page. On this page, the first four columns in the table manage function security, which applies to the object as a whole:

- **Create**
  Users with the corresponding role can create a record of the object.

- **View**
  Users with the corresponding role can view the object’s work area pages.

- **Update**
  Users with the corresponding role can update a record of the object.

- **Delete**
  Users with the corresponding role can delete a record of the object.
The next three columns in the table manage data security.

- **View All**
  Users with the corresponding role can view the object’s records.

- **Update All**
  Users with the corresponding role can update the object’s records.

- **Delete All**
  Users with the corresponding role can delete the object’s records.

To let users view, update, or delete records at run time, you must enable both function security as well as data security for an object. To let users create records, you only have to enable function security.

**Tip:** When clicking **View All**, **Update All**, or **Delete All**, the corresponding **View**, **Update**, and **Delete** function security check boxes are automatically selected. Wait for the page to refresh to confirm all your selections.

### Custom vs. Predefined Roles

The Define Policies page (for both custom objects and roles) displays custom roles. Custom roles are copies of the predefined roles that Oracle provides for all customers. You can’t modify predefined roles, so they aren’t displayed here. However, you can modify custom roles. Modifying a custom role means adding access to one or more custom objects so that role can view the custom object at run time.

If you don’t see a list of roles on the Define Policies page, then you must first copy the predefined roles that you need using the Security Console:

1. Use the Security Console to make copies of the predefined roles you need. These copied roles are known as custom roles.
   - In the Oracle Sales Cloud Securing Oracle Sales Cloud guide, see:
     - Copying Sales Roles: Points to Consider
     - Copying Job or Abstract Roles: Procedure

2. Navigate back to Application Composer, open the Security node for your custom object, then define the security policy across roles for your custom object.

If you upgraded from a previous release of Oracle Sales Cloud, then you might have made changes to predefined roles in an earlier release. During the upgrade to the current release, Oracle automatically copies those modified predefined roles for you, so they will appear as custom roles on the Define Policies page. See "Custom Roles and the Upgrade Process: Explained" in the Oracle Sales Cloud Securing Oracle Sales Cloud guide.

### Application Composer and the Security Console

The Security Console manages the security policies that control access based on roles. However, you define the security policies for custom objects in Application Composer’s object-centric and role-centric Define Policies pages. This is outside the Security Console.

Security policies defined in Application Composer can be modified in Application Composer. Do not use the Security Console to modify these policies.

**Related Topics**

- Enabling Sales Administrators to Test Configurations in the Sandbox
• Assigning Yourself Additional Job Roles Required for Testing

• Copying Sales Roles: Points to Consider

• Copying Job and Abstract Roles: Procedure

Managing Security by Object: Explained

When you create custom objects, by default their UIs are visible only if you have the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. The UI pages you create for custom objects are not visible to additional users unless you provide access in Application Composer using the object’s Security node. Provide access to a single custom object, across multiple custom roles, using each custom object’s Security node. You can specify the job roles that can access the UIs, as well as the levels of access. For example, you can make it possible for sales managers to update object records, while sales representatives can only view records.

Alternatively, you can update the security policy for a custom role, across multiple custom objects, using the Role Security link in the Common Setup pane. See "Managing Security by Role: Explained."

Managing Object Security

The object-centric Define Policies page displays a list of the custom Oracle Sales Cloud roles available for selection. Use this page to manage access to either a top-level or child custom object by specifying a security policy for one or more custom roles. When you do this, users with the corresponding custom roles can access the custom object and its data, depending on the security policies you define.

To access the object-centric Define Policies page:

1. Ensure that you’re in an active sandbox session.
2. Navigate to Application Composer and on the main Overview page, select a custom object in the object tree.
3. Select the Security node. The page that displays is the object-centric Define Policies page.

From the object-centric Define Policies page, you can:

• Enable function security across multiple roles.
• Enable data security across multiple roles.

See "Making Custom Object Pages Visible to Users: Explained" to learn about function security and data security.
Managing Security by Role: Explained

When you create custom objects, by default their UIs are visible only if you have the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. The UI pages you create for custom objects are not visible to additional users unless you provide access in Application Composer. Provide access to a single custom role, across multiple custom objects, using the Role Security link in the Common Setup pane. Use this link to specify not only which job roles can access the UIs, but the levels of access. For example, you can make it possible for sales managers to update object records, while sales representatives can only view records.

Alternatively, you can update the security policy for a custom object, across multiple custom roles, using each custom object's Security node. See "Managing Security by Object: Explained."

Managing Role Security

The Role Security page displays a list of the custom Oracle Sales Cloud roles available for selection. Select a custom role and click the Define Policies button to navigate to the role-centric Define Policies page, which displays a list of the custom objects for your Sales Cloud implementation. Use this page to manage access for users with the corresponding custom role by specifying a security policy for one or more top-level or child custom objects. When you do this, users with the corresponding custom role can access the custom objects and related data, depending on the security policies you define.

To access the role-centric Define Policies page:

1. Ensure that you're in an active sandbox session.
2. Navigate to Application Composer and in the Common Setup pane, select the Role Security node.
   - Or, select the Role Security hyperlink in the local area of the main Overview page.
   - Or, from the object-centric Define Policies page, select a role.
3. Click the Define Policies button. The page that displays is the role-centric Define Policies page.

From the role-centric Define Policies page, you can:

- Enable function security across multiple objects.
- Enable data security across multiple objects.
See "Making Custom Object Pages Visible to Users: Explained" to learn about function security and data security.

- View related roles, if any.

If a related role is displayed next to an object, then the selected role is inheriting its access to that object from the related role. You can drill down into the related role to view its security policies.

### Working with Fields

#### Defining Fields: Explained

Using Application Composer, you can extend Oracle Sales Cloud by adding new fields to both standard or custom objects. The fields that you add to an object are custom fields. When creating a custom field, Application Composer provides a set of field types that you can choose from. For example, you can create a check box field, or create a long text field.

#### Viewing an Object's Fields

A Sales Cloud object can have a maximum of 625 fields. To review the standard and custom fields for an object, and to create custom fields, navigate to the object’s Fields page in Application Composer.

1. Navigate to Application Composer from the Configuration category in the Navigator.
2. Expand the object that you want to add custom fields to.
3. Select the Fields node to navigate to the Fields page.

Click the Standard Fields or Custom Fields tabs to view the standard or custom fields available for the object.

- **On the Standard Fields tab:**
  
  Review the list of standard fields that are delivered for an object, and optionally modify the display label and help text for a field.
  
  The list of standard fields includes all the fields that are delivered by Oracle for an object, as well as system fields, which could include:
  
  - CreatedBy
  - CreationDate
  - Id
  - LastUpdateDate
  - LastUpdatedBy
  - RecordName

  **Note:** The custom objects that you create also contain these same system fields, among others.

- **On the Custom Fields tab:**
  
  Review the list of custom fields that you created specifically for your Sales Cloud implementation, and create new custom fields.
Adding Fields to Objects

To create a custom field:

1. Confirm that you’re in a sandbox session, before making any changes to Oracle Sales Cloud.
2. In Application Composer, select the object that you want to make changes to, then select the object’s Fields node.
3. On the Custom Fields tab, click New.

Application Composer provides a set of field types that you can choose from when creating new fields:

- Check box
- Currency
- Date
- Datetime
- Dynamic choice list
- Fixed choice list
- Formula
- Long text
- Number
- Percentage
- Record Type
- Text

4. Select the type of field you want to create, and then specify the required field attributes to create the custom field.
5. After you create custom fields, you must expose those fields on the right user interface pages, before your end users can see them. See Defining Pages: Explained.

When you create custom fields for objects and expose the fields on desktop pages, Application Composer automatically creates all the underlying object artifacts for you, and provides full Web service support for those new fields, as well. Application Composer also makes it easy to enable your object model extensions for importing and exporting.

Deleting Fields

You can’t delete either standard or custom fields from objects. If you no longer need a field that was already published to the mainline metadata, optionally enter a note in the field description that the field is no longer used.

Related Topics

- Defining Pages: Explained
- Changing Field Display Labels: Explained

Field Types and Field Properties: Explained

In Oracle Sales Cloud, fields can be one of several types, such as a number field or formula field. And, each field type has a set of standard properties. Most properties are common across all types of fields, although some are specific to the type of field you’re creating. For example, all field types have a display label, but only some field types have a display width specification. Read this topic to learn about field types and common field properties.
Before you create a new field for an object, you should understand:

- The field types available for field creation
- The common set of field properties that you can specify for a field
- How field types work with other components
- Extensible properties for standard fields

Field Types

Application Composer provides a set of standard field types that you can choose from when creating a new field for an object.

The types of fields that you can create are:

- **Check box**
  Select to indicate a true or false attribute of a record.

- **Currency**
  Enter a currency amount.

- **Date**
  Enter a date, or select one from a calendar.

- **Datetime**
  Enter a date, or select a date from a calendar, and enter a time of day. During field creation, you choose whether to show the date or time.

- **Dynamic choice list**
  Select from a list of values populated from another object’s set of records.

- **Fixed choice list**
  Select from a list of static values populated from an FND lookup type.

- **Formula**
  Calculated in the runtime application using the Groovy-based expression included in the formula field’s definition. This is a read-only field that users in the runtime application do not update. However, the application logic that you write can update these fields directly.

- **Long text**
  Enter a combination of letters, numbers, or symbols. This field type supports 32,000 characters.

- **Number**
  Enter a number in this field.

- **Percentage**
  Enter a percentage. The percent sign is automatically added.

- **Record Type**
  Select from a list of static values populated from an FND lookup type. You can associate each choice list value with a role or a page layout.

- **Text**
Enter a combination of letters, numbers, or symbols. This field type is limited to 1500 characters.

Common Properties for Custom Fields

When you create a custom field, you first select the field type. You cannot change the field type after the field is created. The specified field type controls which field properties you must define when creating the field. Some properties are common across field types, while other properties are unique to a specific field type.

The common field properties that you can define for a custom field are listed in this table, along with the regions on the field configuration pages where they appear and a list of the applicable field types that you must set these properties for. Use this table to understand the common properties that you must define when creating a new field.

Tip: Many of these properties are also available for standard fields. See "Common Properties for Standard Fields" in this topic.

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
<th>Related Field Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
<td>Set this property for all field types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify the display label for the field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You should limit the label to a maximum length of 80 characters, although no maximum length is enforced.</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
<td>Set this property for all field types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays when users hover over the field in the runtime application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You should limit the label to a maximum length of 80 characters, although no maximum length is enforced.</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
<td>Set this property for all field types except for check box, date, and datetime fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify for most field types at runtime. The display width is the actual character width for the field on a desktop page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When setting the display width, consider the resolution in use where this field will appear on a desktop page. A display width that is too wide will stretch beyond the resolution of the display and result in scroll bars.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generally, enter a display width of no more than 20 to 25 characters.</td>
</tr>
</tbody>
</table>

Note: For fixed choice lists, note that the display width is determined by the length of the longest string in the choice list.

| Name             | Name                  | Set this property for all field types.                                               |
Field Property

A unique field name for internal use only.

The field name is automatically populated based on the field label you enter, but without spaces.

Field names can contain only underscores and alphanumeric characters. They must:

- Begin with a letter
- Not contain spaces
- Not end with an underscore
- Not contain consecutive underscores
- Not include special characters.
  This might cause issues while generating clients using SOAP web services.
- Be limited to a maximum of 28 characters if the characters are single byte.
  If the characters are multibyte, such as Japanese or Chinese, then the maximum character limit is 28/number of bytes per multibyte character.
  For example, if characters are 2 bytes, then the name is limited to a maximum of 14 characters.
  If a mix of characters is used, then 28 is the maximum sum of character bytes that is supported.

You cannot change this property after the field is created.

Tip: It is possible to create custom fields with different names, but the same display label. Avoid this scenario, however, to avoid seeing two fields with the same display label when configuring a user interface page.

The API name, used in your Groovy scripts, is also automatically generated for a field by taking the logical name and appending \_c. Don’t use special characters in the API name. Also, the API name must be in English. Otherwise you won’t be able to add the field to any page.

Description

A unique field description for internal use only.
<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
<th>Related Field Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Constraints</td>
<td>Set this property for all field types except for formula fields.</td>
</tr>
<tr>
<td></td>
<td>Specify constraints, which let you control the runtime behavior of the field.</td>
<td></td>
</tr>
<tr>
<td>Default values are not necessary for required fields. However, you must expose all required fields on the object’s creation and details (update and edit) pages wherever those pages appear (such as on the desktop, simplified, mobile, or Outlook UI). Required fields are automatically added to an object’s creation pages. However, they are not automatically added to details pages; you must do this manually. This lets your users populate the field at runtime. The object’s web services also reflect the required fields when your sandbox is published to the mainline metadata.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
<td>Set this property for all field types except for formula fields.</td>
</tr>
<tr>
<td></td>
<td>Specify constraints, which let you control the runtime behavior of the field.</td>
<td></td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
<td>Set this property for all field types except for long text and formula fields.</td>
</tr>
<tr>
<td></td>
<td>Specify constraints, which let you control the runtime behavior of the field.</td>
<td></td>
</tr>
</tbody>
</table>
Indexed

Index the field to speed up the performance of saved searches in the different object work areas. The global search is not affected.

Use this option only on the most frequently searched custom fields because, to ensure search performance, the number of fields you can index is limited:

- For standard objects, you can index two text fields and three number fields (shared among number, percentage, and currency fields).
- For custom objects, you can index 10 text fields and 10 number fields (shared among number, percentage, and currency fields).
- Dynamic choice list fields and relationships automatically use 1 indexed number field. If all indexed number fields are already taken, then Application Composer uses a non-indexed number field. If a tab or BI analysis is based on a dynamic choice list field or other relationship, then create that relationship first to ensure you obtain an indexed number field. This ensures optimal performance for your tab or analysis.

You cannot change this property after the field is created.

Include in Service Payload

Specify whether the field value can be included in a web service request or response.

Set this property for all field types.

Fixed Value

Specify a literal default value for the field.

Do not assign a literal default value to fields that are both required and intended to be unique, as a runtime error can occur.

Default Value

Set this property for all field types except formula fields and dynamic choice lists.

Expression

Use the expression builder to write an expression that dynamically sets the default value for a field at runtime.

Default Value

Set this property for all field types except:
- Check Box
- Formula
- Fixed Choice List
- Dynamic Choice List

Fixed Value

Specify a literal default value for the field.

Do not assign a literal default value to fields that are both required and intended to be unique, as a runtime error can occur.

Default Value

Set this property for all field types except formula fields and dynamic choice lists.
Adding Objects and Fields in Application Composer

How Field Types Work with Other Components

When you create new fields for objects, Application Composer limits you to a set of standard field types. The field types that you can select from are already integrated with other components of the Oracle Sales Cloud Extensibility Framework to provide you with the maximum flexibility when extending your Sales Cloud implementation:

- All field types correspond to API data types; each field type has an API name, such as `customfield_c`. When writing a server script using the expression builder, use this `_c` field name to reference fields.
- All field types correspond to your web service XSD payload.
- All field types correspond to your import ODI mappings when using Application Composer’s import and export feature.
- Most field types correspond to available fields that you can use to create a custom subject area for reporting. Exceptions include long text and formula fields.

Extensible Properties for Standard Fields

The extensible field properties that are available for a standard field are listed in this table.

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
</tbody>
</table>

Indicate if the field is required. You can also optionally use the expression builder to write an expression that specifies the conditions that must apply for this field to be required. If a standard field is already set to required by Oracle, however, then you can’t change the field to be not required.

If you write an expression to control whether a field is required, then you must also configure the `Depends On` choice list. This choice list includes fields from the current object, and is used in the evaluation of your expression at runtime.

Default values are not necessary for required fields. However, you must expose all required fields on the object’s creation and details (update and edit) pages wherever those pages appear (such as on the desktop, simplified, mobile, or Outlook UI). Required fields are automatically added to an object’s creation pages. However, they are not automatically added to details pages; you must do this manually. This lets your users populate the field at runtime.
Check Box Fields: Explained

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding fields to both standard or custom objects. One such field is a check box: users in the run time application can select it to indicate a record’s true or false attribute.

Check Box Field Properties

Create a check box field by specifying values for the common set of field properties, such as display label and field name.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
</tbody>
</table>
Oracle Sales Cloud
Extending Sales

Chapter 4
Adding Objects and Fields in Application Composer

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

**Additional Check Box Field Specifications**

Additional specifications for this field type include the following details:

- Data type is VARCHAR2.
- An object can have a total of 625 fields. Of these, 350 are reserved for text and check box fields, and fixed and dynamic choice lists.

**Currency Fields: Explained**

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding new fields to both standard or custom objects. One such field is a currency field, where users in the run time application can enter a currency amount.

**Currency Field Properties**

Create a currency field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the currency field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
</tbody>
</table>
The following properties are unique to certain field types, including currency fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Length</td>
<td>Constraints</td>
</tr>
</tbody>
</table>

Specify how many digits a user can enter in the field.

The maximum length is the total number of digits that the currency field can have. Decimal places are validated against what is configured for the currency code in Setup and Maintenance. Refer to the following section for setup instructions.

<table>
<thead>
<tr>
<th>Minimum Value</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>The minimum numeric value that a user can enter into this field.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Value</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>The maximum numeric value that a user can enter into this field.</td>
<td></td>
</tr>
</tbody>
</table>

**Additional Currency Field Specifications**

Additional specifications for this field type include the following details:

- Data type is NUMBER.
- An object can have a total of 625 fields. Of these, 200 are reserved for number, currency, and percentage fields.

**Note:** Each currency field uses two number type columns: one stores the amount itself, and the other stores the currency conversion rate that is calculated from the entered amount’s currency code to the corporate currency code.
A Sales Cloud object includes the following fields to assist with currency conversion. These fields are automatically added to a Sales Cloud object if the object’s application allows the creation of currency fields. They are derived from the application’s corporate currency setup.

- Currency code
  - The currency code for all of an object’s currency fields.
- Corporate currency code
- The currency conversion rate type.

Currency conversion for a currency field occurs as follows:

- At run time, the user enters the currency amount.
- When the user saves the record:
  - The currency amount is stored using the currency code specified for the object.
  - The application calculates the currency conversion rate using the object’s currency code, corporate currency code, currency conversion rate type, and the currency field’s specified exchange date, if any.

In addition to the entered amount, only the conversion rate that is calculated from the entered amount’s currency code to the corporate currency code is stored.

- If you later change either the currency code or exchange date, the application recalculates the currency conversion rate for the record. Note, however, that the currency amount displayed on the application page will not change.

**Note:** When you run a report based on a custom subject area that uses a currency field, the report does display your preferred currency based on the current exchange rate. Again, this is different from how currency amounts are displayed at run time on application pages, because currency fields only ever display in the entered amount, even if the currency conversion rate for the record changes.

- Precision, or the number of decimal points, for a currency field is derived from the currency code itself. To set the precision for a currency code:
  a. Under Setup and Maintenance, search on “Manage Currencies” on the All Tasks tab, and click **Go To Task**.
  b. In the Currency Code field, enter a currency code, such as JPY.
  c. In the Search Results region, expand the currency code and enter a number into the Precision field.

For example, to display two decimal places for currency fields based on JPY, enter 2 in the Precision field for the JPY currency code.

**Date Fields: Explained**

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding fields to both standard or custom objects. One such field type is a date field, where users in the run time application can enter a date or select one from a calendar. This type of field has no time component.

**Date Field Properties**

Create a date field by specifying values for the common set of field properties, such as display label and field name.
The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

**Additional Date Field Specifications**

Additional specifications for this field type include the following details:

- Data type is TIMESTAMP.
- An object can have a total of 625 fields. Of these, 50 are reserved for date and datetime fields.
- When you create a custom subject area for custom reporting, you can select fields with this type to use for date leveling.

**Datetime Fields: Explained**

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding new fields to both standard or custom objects. One such field type is datetime, where users in the run time application can enter a date or select one from a calendar, and enter a time of day.

**Datetime Field Properties**

Create a datetime field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the datetime field type.

The following properties are common across multiple field types:
### Additional Datetime Field Specifications

Additional specifications for this field type include the following details:

- Data type is TIMESTAMP.
- A object can have a total of 625 fields. Of these, 50 are reserved for date and datetime fields.
- When you create a custom subject area for custom reporting, you can select fields with this type to use for date leveling.
- This field type supports time zone conversion.

### Dynamic Choice Lists

**Dynamic Choice Lists: Explained**

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding new fields to both standard or custom objects. One field type that you can add to a custom or standard object is a dynamic choice list. A dynamic choice list is a field that contains a list of values which are populated from the actual data of another object. For example, you might want to expose on a user interface page a dynamic choice list which lets users specify the account that
they are logging a help request against. In this example, the **Account Name** choice list is a field that you are adding to the help request object, but the list of values is populated by actual names from the account object.

When creating dynamic choice lists, note the following:

- Review the common set of field properties, as well as the dynamic choice list-specific properties, that you must specify.
- Review the options available in the List Data Source, Additional List Display Values, and Additional List Search Fields regions.
- Understand how a dynamic choice list results in the implicit creation of a relationship.

**Note:** You must create a Select and Search dialog box (picker) for a custom object, if you also create a dynamic choice list that is based on the same custom object.

### Dynamic Choice List Properties

Create a dynamic choice list by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the dynamic choice list field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including dynamic choice lists:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Object</td>
<td>List Data Source</td>
</tr>
<tr>
<td>List Selection Display Value</td>
<td>List Data Source</td>
</tr>
<tr>
<td>Data Filter</td>
<td>List Data Source</td>
</tr>
</tbody>
</table>
Using the List Data Source, Additional List Display Values, and Additional List Search Fields Regions

When defining a dynamic choice list, use the following regions to determine what data will display in the list of values at runtime.

**Note:** Although you can configure what data will display in the list of values at runtime, you cannot control the sequence of where those values display in the list.

- List Data Source region

  1. Related Object

     The values in a dynamic choice list are populated from another object’s data. Select the related object first, then use the **List Selection Display Value** choice list to select the field that you want to display within the dynamic choice list as the first column at runtime. Selecting the related object is possible only during field creation.

     **Note:** The set of objects that are available for selection is constrained to top-level objects only. You cannot select a child object as a related object.

     In our example, the related object would be **Account**.
Tip: Once you create a dynamic choice list field, you can easily recognize the choice list’s related object from the Fields page. The Fields page displays summaries of both standard and custom fields for the selected object. If a dynamic choice list was created, then the Type column includes the related object. In our example, the field type would be **Choice List (Dynamic) <Account>**.

- List Selection Display Value

  From the **List Selection Display Value** choice list, select the related object’s field that you want to display within the dynamic choice list as the first column at runtime. This is the primary field on the related object that your users will use to make the appropriate selection. In our example, the field might be something like **Name**.

  **Note:** The fields available for selection include only the related object’s required fields.

- Data Filter

  You can further refine the set of data that appears within the dynamic choice list at runtime by using data filters. You can set up a simple filter to show only a subset of records in the list of values at runtime. Or, you can set up a more complex expression that dynamically filters the subset of records based on the object context. You can also use an existing filter that is predefined for some standard objects.

  Ideally, set data filters on a dynamic choice list during the initial configuration of the field.

  For example, use a simple filter to hide any accounts outside a particular region. Or, hide account records that are inactive. At runtime, only active accounts would appear in the dynamic choice list field.

  **Tip:** To optimize performance, refine the list of values displayed in this field at runtime by including one or more filters based on indexed fields.

  To learn more about data filters, see "Understanding Data Filters for Dynamic Choice Lists: Explained."

- Additional List Display Values region

  Related object fields also included in the List of Values window for context.

  **Available Fields**
  - CreatedBy
  - CreationDate
  - Id
  - LastUpdateDate
  - LastUpdatedBy

  **Selected Fields**
You can further refine the look and feel of the dynamic choice list by selecting additional fields to display in the choice list.

Use the **Additional List Display Values** region to include additional related object fields in the dynamic choice list at runtime. These additional fields assist your users in making a selection from the choice list. The region does not include the field that you already selected in the **List Selection Display Value** choice list.

There is no limit on the number of additional fields that you can select.

- Additional List Search Fields region

You can indicate which additional related object fields can be added as search criteria in the dynamic choice list's Search and Select dialog.

Use the **Additional List Search Values** region to include additional related object fields in the dynamic choice list's Search and Select dialog, accessed using the **Search...** link at runtime. The region does not include the field that you already selected in the **List Selection Display Value** choice list.

There is no limit on the number of additional fields that you can select.

### Implicit Relationship Creation

When you create a dynamic choice list for an object based on a related object, you are implicitly creating a one-to-many foreign key relationship where the current object is the "many" object and the related object is the "one" object. This implicit creation of a relationship lets you later add a related object subtab for the "many" object on the "one" object's details page. You can view these implicitly created choice list relationships on the Relationships page.

In the previous example of making a list of accounts available for selection for a help request, an account can be tied to multiple help requests. The relationship that is created is a one-to-many relationship between the account and help request objects, which enables users to store an account identifier in the help request object’s table. In this relationship, the account object is the source object and the help request object is the target object. If a source object is ever deleted, then at runtime, then the dynamic choice list will have no values in it.
Later, you might want to expose a related object subtab on the account details page which shows, for a single account, all the help requests that are related to it. You can create this related object subtab because the relationship was already created when you created the dynamic choice list.

**Note:** Generally, the dynamic choice list that you create results in the implicit creation of a choice list relationship. The exception is if you are in a global single instance environment and you create a dynamic choice list between a Sales Cloud object and a common object: resource, customer contact profile, account, address. In such cases, relationships are not implicitly created.

**Groovy Script Syntax**

Once a one-to-many relationship is created between objects using a dynamic choice list field, a foreign key field is created on the “many” object. Use the following API names to access those foreign keys in your scripts.

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Foreign Key API Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic choice list relationship</td>
<td>If the dynamic choice list field name is DynChoice1_c, then the foreign key API name is DynChoice1__Id_c.</td>
</tr>
</tbody>
</table>

**Additional Dynamic Choice List Specifications**

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500).
- A object can have a total of 625 fields. Out of those 625 fields, 350 fields are reserved for text and check box fields, and fixed and dynamic choice lists.
- Dynamic choice list fields and relationships automatically use 1 indexed number field. If all indexed number fields are already taken, then Application Composer uses a non-indexed number field.
  
  If a subtab or BI analysis is based off a dynamic choice list field or other relationship, then create that relationship first to ensure you obtain an indexed number field. This ensures optimal performance for your subtab or analysis.

**Understanding Data Filters for Dynamic Choice Lists: Explained**

Using Application Composer, you can add a dynamic choice list field to a custom or standard object. A dynamic choice list is a field that contains a list of values which are populated from the actual data of another object. You can set up a simple filter on the field to show only a subset of records in the list of values at runtime. Or, you can set up a more complex expression that dynamically filters the subset of records based on the object context. You can also use an existing view criteria filter that is predefined for some standard objects. This topic explains how to use data filters with dynamic choice list fields.

**Understanding Data Filters**

When you first define a dynamic choice list, you indicate which related object’s records populate the custom field’s list of values. You can further refine the set of data that appears within the dynamic choice list at runtime by using data filters. Application Composer lets you define either a simple filter or an advanced filter. Or, use an existing view criteria filter that is predefined for some standard objects. Ideally, set data filters on a dynamic choice list during the initial configuration of the field.

To refine the set of data that appears within the dynamic choice list, you can select from one of three options:

- Simple mode
- Advanced mode
• Existing filter

Tip: When defining a simple or advanced filter, use indexed fields to optimize performance.

Using Simple Data Filters
A simple data filter uses static values that you provide as search criteria to refine the list of values at runtime. For example, when editing an opportunity, the list of accounts that display in the Account dynamic choice list can be filtered to show only active accounts.

To define a simple data filter:

1. In Application Composer, create a custom dynamic choice list field.
2. After you select the related object and the object’s field that you want to display in the list of values, navigate to the Data Filter region.
3. In the Data Filter region, click Add Search Field.
4. Select the field that you want to use as the simple filter, then specify the criteria to apply at runtime.
   - For example, if you select City, then specify a city name, such as New York.
   - You can select more than one field.
5. Complete the rest of the field’s configuration, then click Submit.

The filter’s static value is applied to the set of records in the dynamic choice list at runtime to refine the list of values.

Using Advanced Data Filters
An advanced data filter uses dynamic criteria based on the object context to refine the list of values at runtime. As part of the advanced data filter, you can write Groovy expressions and accept bind variables.

To define an advanced data filter:

1. In Application Composer, create a custom dynamic choice list field.
2. After you select the related object and the object’s field that you want to display in the list of values, navigate to the Data Filter region.
3. In the Data Filter region, click the Advanced Mode link that displays in the instructions.
4. Enter a search expression to refine the records that appear in the list of values. As you write your script:
   - Click Add Search Field to select appropriate fields.
   - Click Add Bind Variable to pass the target object context to the filtering expression.
   - For example, when editing an opportunity, the list of contacts that display in the Contact dynamic choice list can be filtered by adding bind variables that pass context from the opportunity record to the Contact dynamic choice list.
5. Check the When filtering data, ignore expressions involving null bind variable values check box.
   - This check box defines the behavior of the expression when a bind variable evaluates to null. If this box is checked, then any atomic expression ("FieldName Operator Operand") whose bind variable evaluates to null is ignored. The other atomic expressions that are part of the filter remain.
   - For example, when editing an opportunity, the list of contacts that display in the Contact dynamic choice list should display only the contacts of the opportunity record’s specified account. However, if there is no account specified in the Account field, then all contacts should be displayed in the list of values. To achieve this, check the When filtering data, ignore expressions involving null bind variable values check box. If the box is not checked and no account is specified, then the Contact dynamic choice list will be empty.
6. Complete the rest of the field’s configuration, then click Submit.
Using Existing View Criteria Filters
When defining a data filter for a dynamic choice list, you can select from a predefined list of existing view criteria filters. These existing filters are the named view criteria that are already provided for some standard objects. Some standard objects may predefine named view criteria which you can use to simplify common searches.

To select an existing view criteria filter:

1. In Application Composer, create a custom dynamic choice list field.
2. After you select the related object and the object’s field that you want to display in the list of values, navigate to the Data Filter region.
3. In the Data Filter region, click the Existing Filter link that displays in the instructions.
4. Select the desired filter.
5. Complete the rest of the field’s configuration, then click Submit.

Fixed Choice Lists: Explained
Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding new fields to both standard or custom objects. One such field type is a fixed choice list, a field that contains a list of static values populated from FND lookup types. At run time, users can select one or more values from this field, depending on the field’s definition.

Fixed Choice List Properties
Create a fixed choice list by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the fixed choice list field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
</tbody>
</table>
Fixed Value

You cannot set a default value for any fixed choice list that is constrained by another fixed choice list.

If the choice list allows multiple values, you can still write an expression to preselect multiple values by default.

For example, if the field includes three lookup codes with (Code,Label) of (S,Small),(M,Medium),(L,Large), and (XL,Extra Large), then to preselect the Small and Extra Large selections by default, set the default value to the literal string (without quotes): S,XL.

The data for the multi-select pick list is stored in comma-separated format; in the previous example, “S,XL” will be stored in the database.

The following properties are unique to certain field types, including fixed choice lists:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Type</td>
<td>Appearance</td>
</tr>
<tr>
<td>Indicate if users can select a single value or multiple values from the choice list at run time. You can only select the display type during field creation.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If you create a multiple-select fixed choice list, then do not use commas in the lookup codes that populate this field.

<table>
<thead>
<tr>
<th>Lookup Type</th>
<th>List of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>You cannot create a Lookup Type with a name ending in &quot;LOOKUPTYPE&quot;. If you do, you won’t be able to see this extension in BI Answers and reporting.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constrain List by Parent Field Value Selection</th>
<th>List of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Selecting the List of Values for the Fixed Choice List

The values in a fixed choice list are populated from FND lookup types. Select the lookup type with values you want to display in this choice list. You can only select the lookup type during field creation. A fixed choice list can have a maximum of 1,000 values.

You can also create a new lookup type and add new values to it. Enter a lookup type and select the Edit icon to modify the existing values.

The set of FND lookup types available for selection is constrained to those lookup types related to this fixed choice list's object (by product code), as well as all custom lookup types that have been created for your Sales Cloud implementation.

Mapping List Values to Parent Values

You can constrain the actual values that display in the fixed choice list at run time by relating the fixed choice list to a parent fixed choice list. The value selected in the parent fixed choice list drives the values that display in this fixed choice list.

For example, you might want your users to see two choice lists on a desktop page where they can create a trouble ticket: one for specifying the trouble ticket type and one for specifying the trouble ticket area. If a user selects Hardware from the Type choice list, then the Area choice list should contain only hardware options that the trouble ticket can be logged against, such as Desktop or Laptop.
To do this, while creating the **Area** fixed choice list, select the **Constrain List by Parent Field Value Selection** check box, select the **Type** parent field, and then map the values between the parent lookup type and this field’s lookup type.

To implement the previous example:

1. Define the **Type** fixed choice list.
2. Define the **Area** fixed choice list.
   
   a. Select the **Constrain List by Parent Field Value Selection** check box and select the parent field, **Type**.

   You can select the **Constrain List by Parent Field Value Selection** check box only during field creation, and only if at least one other single-select fixed choice list has been defined. After field creation, however, you can update the mapping between lookup values.

   b. Map the values between the Type and Area lookup types.

   For example, map all relevant hardware values in the Area lookup type’s set of values, such as Desktop and Laptop, to the value of **Hardware** in the Type’s lookup type.

   After your users start using these fields to enter data, don’t change a lookup type’s lookup code values. For example, don’t change `LAPTOP_CODE` to `LAPTOP_CODE1`. If you change a lookup type’s lookup code values, then you will need to manually re-map all records that already reference the original code, as well as re-map the values between lookup types in the child fixed choice list.
Note: You cannot set a default value for any fixed choice list that is constrained by another fixed choice list.

Additional Fixed Choice List Specifications
Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500).
- An object can have a total of 625 fields. Of these, 350 are reserved for text and check box fields, and fixed and dynamic choice lists.

Related Topics
- Lookups: Explained

Formula Fields: Explained
Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding fields to both standard or custom objects. One such field is a formula field, which is calculated in the run time Sales Cloud application using the Groovy-based expression included in the field’s definition.

Formula Field Properties
Create a formula field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the formula field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
</tbody>
</table>

The following properties are unique to certain field types, including formula fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula Type</td>
<td>Field Value Type</td>
</tr>
</tbody>
</table>

Specify the field’s data type, such as text, number, or date. You can specify the type only during field creation.
Using the Expression Builder and the **Depends On** Choice List

Use the **Depends On** choice list to indicate if the field should be automatically recalculated (using the expression you write) if another field’s value changes.

> Note: The **Depends On** choice list includes a list of fields that belong to the same object. If you want this formula field to automatically recalculate if the value of another field on a different object changes, then you must write a server script.

Use the expression builder to write an expression that calculates the field’s value at run time.
This list provides several examples on when to use the expression builder:

- If your expression calculates the value of an employee's annual bonus amount, set the expression to automatically recalculate that amount if the employee's salary changes.

- If your expression determines the correct customer phone number to use for an opportunity, set the expression to automatically reset the phone number if the opportunity's customer account changes.

**Additional Formula Field Specifications**

Additional specifications for this field type include the following details:

- Data type is set by the **Formula Type** property.

- The formula field type is not supported by custom subject areas. You cannot add formula fields to a custom report.

- You cannot search on a formula field.
• A formula field is a computed attribute, and exists only at run time. This is a transient type of attribute that does not persist in the database as a table column. Hence, no maximum number of formula fields exists for an object.

• A formula field’s groovy script is evaluated every time the field’s value is requested by any layer. You should not use a formula field to set other fields’ values because, due to the order of rendering, the order in which the fields are processed is not guaranteed. If you want to write code that derives other field values when the value of some other field is changed, use the After Field Changed trigger documented in the Groovy Scripting Reference for Oracle Sales Cloud.

Related Topics
• Groovy Scripting: Explained

Joins and Join Fields: Explained

A join is a predefined association between an object and its related object. Joins use underlying, preexisting relationships already delivered with Oracle Sales Cloud. You use joins to add related object fields to an object’s work area. Before you can do that, however, you must register those fields, either custom or standard, by creating join fields. (Join fields aren’t provided automatically for you.)

Understanding Joins

Joins are view links between an object and another top-level object, which are already related through an existing many-to-one or one-to-one relationship. Joins let you display a related object’s fields on an object’s work area.

Note: You can’t create joins or edit existing joins.

Why Use Joins?

Joins leverage preexisting relationships between an object and its related object. Joins give you more flexibility than relationships provide. With relationships, you can only add child or related object subtabs to an object’s details page. Joins, on the other hand, let you add related object fields to any page of an object’s work area, not just the details page.

For example, the Account object and the Opportunity object are related objects by default and are already delivered with a join. Let’s say you register the Account’s Primary Phone field as a join field for the Opportunity object. You can now display that field anywhere on the Opportunity work area, such as on the Create Opportunity page or Opportunity landing page.

Tip: The other benefit of joins and join fields is that, at run time, when your end users enter data into the Account Primary Phone field, the data is actually written to the underlying Account object.

Choosing a Join

To view the joins available for an object, expand the object and click the Joins node. For example, expand the Opportunity object, and click the Joins node. Select the desired join row and click Edit to navigate to the read-only Join Specification page, where you can review details about the join.

Joins are delivered by default for some objects in Oracle Sales Cloud. (Not every object has a Joins node.) For example, these objects are delivered with one or more joins:

• Customer Contact Profile
• Forecast Item
• Opportunity
• Partner
• Product Group
• Program Enrollments
• Sales Account

**Note:** The Sales Account object is available for change only to existing Oracle Sales Cloud users from prior releases. If the Sales Account object is read only for you in Application Composer, then this means that you cannot extend the Sales Account object. Instead, use the Account object only.

• Sales Lead Contact

### Registering Join Fields

Joins are delivered without join fields. Before you can add related object fields from a join to an object's work area, you must select the related object fields that you want to display, and then register them as join fields.

To register a join field:

1. Expand an object and click the Joins node.
2. Click the join name to navigate to the Join Fields page, where you can register join fields.

### Working with Join Fields

Once you have registered a related object field as a join field, you can then show or hide those fields on an object's work area by using the configuration pages available from the object's Pages node. There are some caveats about working with join fields, listed below.

- Join fields that are based on a dynamic choice list field aren't exposed as searchable fields in Application Composer. This means that when you configure the local search, regional search, Search and Select dialog, or a context link subtab, join fields based on dynamic choice list fields aren't available for selection.

  However, you can still filter the records that display in an object's summary table by using the Query By Example feature. At run time, click the Query By Example icon on the table's toolbar, and enter a value for the join field column.

- Fields configured for an object as a join field do not appear in file-based import and bulk export.

- Join fields are computed attributes, and exist only at run time. This is a transient type of attribute which does not persist in the database as a table column. Hence, there is no maximum number of join fields for an object.

### Long Text Fields: Explained

Using Application Composer, you can extend an Oracle Sales Cloud application's object model by adding fields to both standard or custom objects. One such field type a long text field, where users in the run time application can enter a combination of letters, numbers, or symbols. This field type supports 32,000 characters.

### Long Text Field Properties

Create a long text field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the long text field type.

The following properties are common across multiple field types:
The following properties are unique to certain field types, including long text fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Type</td>
<td>Appearance</td>
</tr>
</tbody>
</table>

Indicate how you want this text field to render in the run time application:

- As a simple text box.
- Allowing multiple lines where text can wrap, or where the user can enter carriage returns.

Additional Long Text Field Specifications

Additional specifications for this field type include the following details:

- Data type is CLOB.
- A object can have a total of 625 fields. Of these, 25 are reserved for long text fields.
- The long text field type is unavailable for use with custom subject areas and Oracle Social Network:
  - Long text fields are not available for inclusion in custom reports.
  - Long text fields are not available for sharing in OSN conversations.
Number Fields: Explained

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding fields to both standard or custom objects. One such field is a number field, where users in the run time application can enter a number.

Number Field Properties

Create a number field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the number field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to certain field types, including number fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal Places</td>
<td>Constraints</td>
</tr>
</tbody>
</table>

The number of digits that can be entered and displayed to the right of the decimal point. If at run time, a user enters more digits after the decimal
Field Property | Field Property Region
---|---
point, then Application Composer rounds up (using the tie-breaking rule, round half up) to derive the field’s value.

For example, if you enter 2 for the number of decimal places, then at run time, an entry of 4.986 is displayed as 4.99.

### Maximum Length

The number of digits a user can enter in the field. This number should be greater than or equal to 1 and less than or equal to 38.

During field creation, consider how this property interacts with these other field properties:

- **Display Width**
  If you set a maximum length that is longer than the display width, then users must scroll inside the field at run time to see the number in this field.

- **Decimal Places**
  Maximum Length - Decimal Places = the number of digits that can appear to the left of the decimal point.
  Do not set a maximum length that is shorter than the number of decimal places.

### Minimum Value

The minimum numeric value that a user can enter into this field.

### Maximum Value

The maximum numeric value that a user can enter into this field.

### Additional Number Field Specifications

Additional specifications for this field type include the following details:

- Data type is NUMBER.
- An object can have a total of 625 fields. Of these, 200 are reserved for number, currency, and percentage fields.
Note: Of the 200 reserved for number, currency, and percentage fields, 10 fields are reserved for indexed number fields. This means you can create a total of 190 non-indexed fields.

- Leading zeros are removed.

Percentage Fields: Explained

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding fields to both standard or custom objects. One such field type is a percentage field, where users in the run time application can enter a percentage. Application Composer automatically adds the percent sign to the number.

Percentage Field Properties

Create a percentage field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the percentage field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including percentage fields:
Field Property | Field Property Region
--- | ---
Decimal Places | Constraints

The number of digits that can be entered and displayed to the right of the decimal point. If at run time, a user enters more digits after the decimal point, then Application Composer rounds up (using the tie-breaking rule, round half up) to derive the field’s value.

For example, if you enter 2 for the number of decimal places, then at run time, an entry of 4.986 is displayed as 4.99.

Maximum Length | Constraints

The maximum digits a user can enter in the field.

During field creation, consider how this property interacts with these other field properties:

- **Display Width**
  If you set a maximum length that is longer than the display width, then users must scroll inside the field at run time to see the amount in this field.

- **Decimal Places**
  **Maximum Length - Decimal Places** = the number of digits that can appear to the left of the decimal point.
  Do not set a maximum length that is shorter than the number of decimal places.

---

Additional Percentage Field Specifications

Additional specifications for this field type include the following details:

- Data type is NUMBER.
- A object can have a total of 625 fields. Of these, 200 are reserved for number, currency, and percentage fields.
- Application Composer automatically adds the percent sign.

Record Type Fields: Explained

Using Application Composer, you can extend Oracle Sales Cloud by adding new fields to both standard or custom objects. One type of field that you can add is a record type field. A record type field is a field that contains a list of static values which
are populated from FND lookup types. This type of field is useful, because you can associate each choice list value with a role or a page layout. This makes a record type field more powerful than a fixed choice list field or a dynamic choice list field.

**Using Record Type Fields**

Create a record type field, so that you can associate each choice list value with a role or a page layout.

You can:

- Associate each choice list value with a role.
  
  a. You do this while you are creating the field.
  
  b. At run time, when an end user views the list of values in the field, the available choices are constrained according to the user’s role.

  Custom roles, which are copies of the predefined roles that Oracle provides for all customers, are displayed by default. However, you can optionally choose to display predefined roles, as well.

- Associate each choice list value with a page layout.
  
  a. You do this by adding the field to a simplified page layout, after you have created the field.
  
  b. You must then assign a choice list value to the page layout.
  
  c. At run time, when an end user selects a value from the field, the page display changes to match the simplified page layout that you associated with the choice list value.

**Note:** You can create only one record type field per object, and only once the object has a work area. If the object’s work area hasn’t been created yet, then you must create the work area first, before creating the record type field.

**Record Type Field Properties**

Create a record type field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the record type field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
</tbody>
</table>

**Note:** The size of the field depends on the longest value of the strings in the choice list.
### Field Property

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Name</td>
</tr>
<tr>
<td><strong>Required</strong></td>
<td>Constraints</td>
</tr>
<tr>
<td><strong>Updatable</strong></td>
<td>Constraints</td>
</tr>
<tr>
<td><strong>Searchable</strong></td>
<td>Constraints</td>
</tr>
<tr>
<td><strong>Fixed Value</strong></td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including record type fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lookup Type</strong></td>
<td>List of Values</td>
</tr>
</tbody>
</table>

Selecting the lookup type is possible only during field creation.

#### Available Record Types

Indicate the choice list values that each role has access to.

For example, perhaps the sales representative can see only selected choice list values, but the sales manager can see all the choice list values.

#### Default Record Type

Indicate the choice list value that each role will see by default at run time.

### Using the List of Values Region

The values in a record type field are populated from FND lookup types. Select the lookup type whose values you want to display in this choice list.

You can also select a lookup type and select the Edit icon to modify the existing values.

**Note:** The lookup types available for selection are limited to:

1. Standard lookup types that are related to this record type field’s object (by product code).
2. All custom lookup types that have been created for your Sales Cloud implementation.

Or, create a new lookup type and add new values to it.
Additional Record Type Field Specifications

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500).
- A record type field is optional, and is not required for an object.
- One record type field is allowed per object, and it will be one of the 350 fields reserved for text and check box fields, and fixed and dynamic choice lists.

Related Topics

- Dynamic Page Layouts: Explained
- Controlling the Display of a Page Based on a Field Value: Explained

Text Fields: Explained

Using Application Composer, you can extend an Oracle Sales Cloud application’s object model by adding new fields to both standard or custom objects. One such field type is a text field, where users in the run time application can enter a combination of letters, numbers, or symbols.

Text Field Properties

Create a text field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the text field type.

The following properties are common across multiple field types:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
</tr>
<tr>
<td>Updatable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Field Property</td>
<td>Field Property Region</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including text fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Type</td>
<td>Appearance</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>Constraints</td>
</tr>
</tbody>
</table>

The way you want this text field to render in the run time application:

- As a simple text box.
- Allowing multiple lines where text can wrap or where the user can enter carriage returns.

**Note:** The 1500-character limit applies if the characters are single byte. If the characters are multibyte, such as Japanese or Chinese, then the maximum character limit is 1500 characters divided by the number of bytes per multibyte character. For example, if characters are 2 bytes, then the name is limited to a maximum of 750 characters. If a mix of characters is used, then 1500 is the maximum sum of character bytes that is supported.

<table>
<thead>
<tr>
<th>Minimum Length</th>
<th>Constraints</th>
</tr>
</thead>
</table>

The minimum number of characters that a user can enter into the field.
Additional Text Field Specifications

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500 char).
- A object can have a total of 625 fields. Of these, 350 are reserved for text and check box fields, and fixed and dynamic choice lists.

Actions and Links: Explained

In Oracle Sales Cloud, you can add actions, such as buttons and menu items, to detail pages, list pages, and so on. You can also create special fields, rendered as links, that are displayed with other fields throughout the application.

Actions and Links

You can base an action on a script (a Groovy method that is defined on the object) or on a URL defined by a script. After you create an action, it can be exposed as a button or an option on the Actions menu. A button can perform an action or navigate the user to another page in the runtime application, or to another Web site. For example, you might want to include a button on a summary table, which users can click at runtime to create a new type of record from a selected row, such as escalating an existing "trouble ticket" to a more severe "case" that can be managed separately.

After you create a link, it can be selected as a field for display at runtime. For example, you might want to provide a static link from an overview page to a corporate Web site.

Adding Actions or Links: Overview

You add actions or links in two steps:

1. Define an action or link for an object.
2. Use Application Composer’s work area configuration pages to add that action or link to a user interface page, such as a list page or details page.

You can also manage the Actions menu by hiding or showing menu items, rearranging the action groupings or display sequence, and managing the toolbar by hiding or showing icons and buttons. You can also configure the Actions menu and buttons in Create and Edit subtabs.
The following figure shows a button and a link added to the Sales Opportunities Overview page.

![Diagram of Sales Opportunities Overview page]

1. **Defining Actions or Links**

To define an action or link for an object:

1. On the main Overview page in Application composer, select a standard or custom object in the object tree.
2. Select the **Actions and Links** node.
3. In the Create Action or Link page, enter a descriptive name in the Display Label field.
4. In the Type field, select either **Action** or **Link**.
5. In the Source field, select either **Script** or **URL**.
6. In the Script region click the **New** icon to build your script.

   - If the source is a URL, you can enter a static URL enclosed in double quotation marks. Or, you can define the URL by using the expression builder, which provides access to this object’s fields to assist you in constructing the URL. If this object has a parent or relationship with a source object, then optionally change the context to access another object’s fields for URL definition.

     Any new functions that you create will be added to the **Method Name** choice list. If functions were already created for the object, then you can select one of them from the Method Name choice list. Object functions that are created elsewhere through other flows, such as server scripts, can also be used here.

     To switch the context to the object’s parent or related source object, for access to the object’s fields for the URL definition, check the **Select alternative context** check box.

   - If the source is a script, you can either select a predefined object function from the **Method Name** choice list, or create a new object function using the expression builder. Any new functions that you create will be added to the **Method Name** choice list.

     If functions were already created for the object, then you can select one of them from the **Method Name** choice list. Object functions that are created elsewhere through other flows, such as server scripts, can also be used here.

     When creating custom actions based on a script for top-level custom objects, you can specify how you would like the action to conclude at runtime, after the script completes:

     - Save the record and return to the previous page (save and close)
     - Save and continue editing the record (save and continue)
Perform the action but don’t save the record (run the script only)

Points to consider when defining actions:

- If you define a custom action and expose it on a list, ensure that you include a check for active record row, and that the UI supports users selecting any record as the active row before invoking the custom action.
- Do not create custom buttons to populate the mandatory or required fields on the UI. End users must enter the values in the mandatory fields manually.

2. Exposing Actions or Links on Pages

After you save actions or links, you can expose them on UI pages by configuring Application Composer options available in the Edit Summary Table page in the Pages node of an object.

When displaying a link, you select it just as you select to display standard or custom fields. This is because, at runtime, the UI displays the URL link as if it is a field in a table.

For example, the following figure shows a link and other fields that are already selected for display in the Edit Summary Table page in the Pages node for the Opportunity object.

Actions can be configured in potentially two places in the UI: on the toolbar as a button and in the Actions menu for a table.

The figure below shows the Configure Summary Table: Actions region, with options checked for the Show Create, Show Edit, and Show Delete options on the Action menu. You can also display the action, Ask_Assistance, as either a button or action menu item.
The following figure shows an overview page with exposed Create, Edit, and Delete options and a custom Ask_Assistance option on the Actions menu. It also shows the custom toolbar button Ask_Assistance, and a custom table column with a custom link.
Tip: To support functions that don’t need to be displayed prominently on the page, add actions as options on the Actions menu. To support key functions that are frequently executed by your users, add actions as buttons. When displaying actions as buttons, be sure to test your page at runtime (in all supported languages) to confirm that the presentation of buttons is as expected. Button display could be unexpected due to the available space on the page at runtime, the number of buttons on the page, and button width (which depends on label length). If you add more buttons than the toolbar has space, then at runtime the buttons are stacked and made available using a drop-down button.

You can display actions as either buttons or Actions menu items in a variety of locations:

- Simplified pages
- Summary table on the overview page
- Default summary on the details page
- Summary table on a details page’s subtab
- Revenue table on the details page for the opportunity object

Note: If you create a custom button on a table that appears on a simplified UI page, and that table has no rows, then the button is automatically disabled.

You can display links for an object in a variety of locations in that object’s work area. You can add a link wherever you can add a field. Possible locations include, but are not limited to:

- As a column in the summary table on the overview page
- Default summary on the details page
- As a column in the summary table on a details page’s subtab
- In the detail form under the summary table on a details page’s subtab
- As a column in the summary table on a tree node page for a child object
- As a column in the revenue table on the details page for the opportunity object

Deleting Unpublished Actions and Links

You can use Application Composer to delete any unpublished actions and links. Any exposure of the actions (as UI buttons or action menus) and links (as fields) in the same object’s extensible pages or as a detailed subtab under another object’s page are also automatically deleted.

To delete unpublished actions or links for an object:

1. On the main Overview page in Application Composer, expand the standard or custom object whose actions or links you want to delete.
2. Select the Actions and Links node.
Application Composer lists all the actions and links defined for the selected object.

3. Select the action or link that you want to delete and click the delete icon.
4. Click **OK** on the confirmation dialog.
5. To verify that the deleted actions and links no longer appear in the object’s pages, click the Pages node and review the page layouts.

**Related Topics**
- Creating and Adding Custom Links to Simplified Pages: Worked Example

### Direct Page Links: Explained

Direct page links are links that point to a specific page. In any e-mail, report, or user interface page, you can add a link that opens an Oracle Sales Cloud account, contact, household, opportunity, lead, activity, or top-level custom object record. When linking to a simplified page, the link opens a specific tab at the top level of an object’s simplified set of pages.

This topic covers:
- List of objects that support direct page linking to both desktop and simplified pages.
- URL pattern to use for direct page linking.
  The URL pattern is translated by the Direct Page Link servlet in the middle tier which reads the incoming request parameters, generates a new URL, and redirects the request to the target page.
- Where can you use these links?
- How user authentication provides secured access to the target page.

### Objects That Support Direct Page Linking

The objects that support direct page linking differ depending on whether you link to simplified or desktop pages.

You can link to the simplified pages for the following objects. The link that you build opens a tab at the top level of an object’s simplified set of pages. Some objects support linking directly to a subtab.

- Account
- Activity
- Contact
- Household
- Lead
- Opportunity
- Custom object
- Partner
- Deal registration
- Fund request
- Claim

> **Note:** You can also link directly to the simplified UI dashboard and to the Analytics landing page.
You can link to the desktop details pages for the objects listed below. The details page is part of an object’s work area where you view the details about a record.

- Contact
- Customer
- Lead
- Opportunity

**URL Pattern to Use for Direct Page Links to Simplified Pages**

The direct page link URL uses a simple pattern which points to a default tab at the top level of an object’s simplified set of pages. Some objects support linking directly to a subtab.

Depending on the object you are linking to, use the following syntax to create a direct page link:

```
https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=<CARDCODE>&fndTaskItemNodeId=<TABCODE>&fnd=%3B<TaskFlowParamName1>%253D<TaskFlowParamValue1>
%253B<TaskFlowParamName2>%253D<TaskFlowParamValue2>%253D<TaskFlowParamName3>%253D<TaskFlowParamValue3>%253B%3B
%3B%3Bfalse%3B256%3B%3B%3B
```

**Tip:** Copy the first part of the URL, `https://<hostname>:<port>/<application>/faces/FuseOverview?`, from the customer instance's actual home page. The `taskFlowParamName` and `taskFlowParamValue` pairs are optional and are typically used for record identifiers. In some cases, an additional parameter, `subTabName`, is also included in the URL. Where supported, this additional parameter allows direct links to the specified subtab.

For example:

```
https://slc08yjp.us.oracle.com:10616/sales/faces/FuseOverview?
fndGlobalItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM_CARD
&fndTaskItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM&fnd=%3BsubTabName%253DOverview%253BAccountPartyId
%253D123456%253B%3B%3B%3Bfalse%3B256%3B%3B%3B
```

**Tip:** Direct page links created in a release prior to Oracle Sales Cloud Release 10 used a different link pattern. However, those direct page links created prior to Release 10 will continue to work in Release 10 and later.

Use the patterns specified as follows to link to the simplified pages for these objects:

- Account.

For this object, use this direct page link URL pattern:

```
https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM_CARD
&fndTaskItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM&fnd=%3BsubTabName%253DOverview%253BAccountPartyId
%253D123456%253B%3B%3B%3Bfalse%3B256%3B%3B%3B
```

This direct page link opens the Overview subtab on the Edit Account simplified page.

To link to other Edit Account subtabs, change the value for the parameter `subTabName` to one of the following values for each subtab, such as `&fnd=%3BsubTabName%253DProfile`.

- Overview
Adding Objects and Fields in Application Composer

• Activity.

For this object, use this direct page link URL pattern to open a list of all activities:

https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=ZMM_ACTIVITIES_CRM_CARD&fndTaskItemNodeId=ZMM_ACTIVITIES_ACTIVITIES_CRM%253B%3B%3B%3B%3Bfalse%3B256%3B%3B%3B

To link to a specific activity (appointment or task), append the record’s primary key as follows: https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=ZMM_ACTIVITIES_CRM_CARD&fndTaskItemNodeId=ZMM_ACTIVITIES_ACTIVITIES_CRM &fnd=%3BFunctionCode%253DTASK %253BActivityId%253D<recordID,such as 123456>%253B%3B%3B%3Bfalse%3B256%3B%3B%3B.

You can also link directly to specific types of activities by changing the value for the parameter &fndTaskItemNodeId to one of the following values:

• Link to a list of all tasks:

Use pattern: &fndTaskItemNodeId=ZMM_ACTIVITIES_TASKS_CRM

• Link to appointments (calendar view):

Use pattern: &fndTaskItemNodeId=ZMM_ACTIVITIES_APPOINTMENTS_CRM

• Contact.

For this object, use this direct page link URL pattern:

https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=HZ_FOUNDATIONPARTIES_CONTACTS_CRM_CARD
&fndTaskItemNodeId=HZ_FOUNDATIONPARTIES_CONTACTS_CRM&fnd=%3BsubTabName%253DOverview %253BContactPartyId %253D<recordID, such as 123456>%253B%3B%3B%3Bfalse%3B256%3B%3B%3B

This direct page link opens the Overview subtab on the Edit Contact simplified page.
To link to other Edit Contact subtabs, change the value for the parameter `subTabName` to one of the following values for each subtab, such as `&fnd=%3BsubTabName%253DProfile`.

- Overview
- Profile
- Team
- Opportunities
- Leads
- Assets
- Relationships
- Recommendations
- Notes
- Activities
- Conversations

- Household.

For this object, use this direct page link URL pattern:

```
https://<hostname>:/<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeID=ZCM_CUSTOMERCTRINFRA360_GROUPS_CRM_CARD
&fndTaskItemNodeID=ZCM_CUSTOMERCTRINFRA360_GROUPS_CRM&fnd=%3BsubTabName%253DOverview
%253BHouseholdPartyId%253D<recordID>, such as 123456%253B%3B%3B%3Bfalse%3B256%3B%3B%3B
```

This direct page link opens the Overview subtab on the Edit Household simplified page.

To link to other Edit Household subtabs, change the value for the parameter `subTabName` to one of the following values for each subtab, such as `&fnd=%3BsubTabName%253DProfile`.

- Overview
- Profile
- SalesAccountTeam
- Contacts
- Opportunities
- Assets
- Leads
- Relationships
- Notes
- Activities
- Conversations

- Lead.

For this object, use this direct page link URL pattern:
This direct page link opens the Summary subtab on the Edit Lead simplified page.

To link to other Edit Lead subtabs, change the value for the parameter subTabName to one of the following values for each subtab, such as &fnd=%3BsubTabName%253DNOTES.

- SUMMARY
- CONTACTS
- QUALIFICATIONS
- SALESTEAM
- ACTIVITIES
- RESPONSES
- NOTES
- OPPORTUNITIES
- CONVERSATIONS
- Analytics1
- Analytics2
- Analytics3

- Opportunity.

For this object, use this direct page link URL pattern:

```plaintext
https://<hostname>:<port>/<application>/faces/FuseOverview?
fndTaskItemNodeId=MOO_OPPTYMGMTOPPORTUNITIES_CRM
&fndGlobalItemNodeId=MOO_OPPTYMGMTOPPORTUNITIES_CRM_CARD&fnd=%3BsubTabName%253DQuotes
%253BskipToEditOptyId%253D<recordID,such as 123456>%253B%3B%3B%3Bfalse%3B256%3B%3B%3B
```

This direct page link opens the Quotes subtab on the Edit Opportunity simplified page.

To link to other Edit Opportunity subtabs, change the value for the parameter subTabName to one of the following values for each subtab, such as &fnd=%3BsubTabName%253DNotes.

- Summary
- Quotes
- Contact
- OpptyTeam
- OpptyPartner
- Activities
- Notes
- Assessments
- Leads
Conversations

You can also link directly to Analytics tabs off the Opportunities landing page, by replacing the fndTaskItemNodeid value with these values:

- MOO_OPPTY_ANALYTICS1_CRM
- MOO_OPPTY_ANALYTICS2_CRM
- MOO_OPPTY_ANALYTICS3_CRM

Custom object.

To create a direct link to the default summary tab for a custom object, use this direct page link URL pattern:

```
https://<hostname>:/<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeId=CRM_CUSTOM_CARD_<XXXX>&fndTaskItemNodeId=CRM_CUSTOM_TAB_<XXXX>&fnd=
%3BsubTabName%253DSUMMARY%253BObjectId%253D<YYYY>%253B%3B%3Bfalse%3B256%3B%3B%3B
```

- Copy the first part of the URL, `https://<hostname>:/<port>/<application>/faces/FuseOverview?`, from the customer instance's actual home page.
- Replace `XXXX` with the custom object's API name using all upper case letters including _C. For example, TROUBLE_TICKET_C. Obtain the API name from the object's overview page (click the object's node in the Custom Objects tree in Application Composer).
- Replace `YYYY` with the custom object's primary key in the database.

For example: `https://slc08yjp.us.oracle.com:10616/sales/faces/FuseOverview?fndGlobalItemNodeid=CRM_CUSTOM_CARD_NEWCUSTOMOBJJAN_C`
To create a direct link to a different subtab, replace "SUMMARY" with the subtab's Component ID. This is an automatically generated ID which you can find by viewing the details page layout where the custom subtab exists.

**Subtabs Region**

- **Partner**
  
  For this page, use this direct page link URL pattern:
  
  https://<hostname>:/<port>/<application>/faces/FuseOverview?
  ZPM_PARTNERS_CARD&findTaskItemNodeId=ZPM_PARTNERS&find=partyId%253D100000151014782%253B%3B%3B%3Bfalse%3B256%3B%3B%3B

- **Simplified UI dashboard.**
  
  For this page, use this direct page link URL pattern:
  
  https://<hostname>:/<port>/<application>/faces/FuseOverview?
  fndGlobalItemNodeId=CRM_FUSE_DASHBOARD_CARD&fndTaskItemNodeId=CRM_DASHBOARD%253D123456%253B%3B%3Bfalse%3B256%3B%3B%3B

- **Analytics.**
  
  For this page, use this direct page link URL pattern:
  
  https://<hostname>:/<port>/<application>/faces/FuseOverview?
  fndGlobalItemNodeId=CRM_FUSE_ANALYTICS_CARD&fndTaskItemNodeId=CRM_ANALYTICS%253D123456%253B%3B%3Bfalse%3B256%3B%3B%3B

- **Deal registration**
  
  For this page, use this direct page link URL pattern:
URL Pattern to Use for Direct Page Links to Desktop Details Pages

The direct page link URL uses a simple pattern which points to a particular details page.

Depending on the object whose details page you are linking to, use the following syntax to create a direct page link:

```url
http://<hostname>:<port>/crmCommon/faces/deeplink?ObjType=<object_name>&ObjId=<123456>
```

In this URL, replace `ObjType` with one of the following supported objects:

- Contact.
  
  For this object, use this direct page link URL pattern:
  
  ```url
  ```

- Customer.
  
  For this object, use this direct page link URL pattern:
  
  ```url
  ```

- Lead.
  
  For this object, use this direct page link URL pattern:
  
  ```url
  http://<hostname>:<port>/crmCommon/faces/deeplink?ObjType=Lead&ObjId=99999700000
  ```

- Opportunity.
  
  For this object, use this direct page link URL pattern:
  
  ```url
  http://<hostname>:<port>/crmCommon/faces/deeplink?ObjType=Opty&ObjId=99123456
  ```

- Fund request
  
  For this page, use this direct page link URL pattern:

  ```url
  https://<hostname>:<port>/<application>/faces/FuseOverview?fnd=%3BFundRequestId
  %253D300100089921747%253BsubTabName%253DSUMMARY%3B%3Bfalse%3B256%3B%3B
  %3B&fndGlobalItemNodeId=itemNode_partner_management_mdf&fndTaskItemNodeId=MKT_MDF_CLAIMS_CRM&_afrLoop=170080752950792&_afrMFW=1600&_afrMFH=799&_afrMFDW=1600&_afrMFDH=900&_afrMFC=8&_afrMFCI=0&_afrMFM=0&_afrMFR=96&_afrMFG=0&_afrMFS=0&_afrMFO=0
  ```

- Claim
  
  For this page, use this direct page link URL pattern:

  ```url
  https://<hostname>:<port>/<application>/faces/FuseOverview?fnd=%3BClaimId
  %253D300100089921736%253BsubTabName%253DSUMMARY%3B%3Bfalse%3B256%3B%3B
  %3B&fndGlobalItemNodeId=itemNode_partner_management_mdf&fndTaskItemNodeId=MKT_MDF_FUNDS_CRM&_afrLoop=16917244458
  ```
Assigning These Links

First, you should know which page you want to link to, and where you want that link to appear. You can add direct page links to:

- **Reports**
  Use BI Composer or BI Answers to add direct page links to your reports.
- **User interface pages**
  Create a direct page link using an object’s Actions and Links node, then add the link to the object’s user interface pages.
- **External third-party applications**
  Create a direct page link to link directly to a page in Oracle Sales Cloud.

User Authentication for Secured Access

The direct page link servlet requires authentication. If you have not been previously authenticated, you must log in to gain access to the target page. After login, you are redirected to the target page. If you have already been authenticated at the time of clicking the direct page link, the target page is immediately displayed (without asking you to login).

Importing and Exporting Custom Objects: Explained

Before you can import and export data for custom objects created with Application Composer, you must first generate the object artifacts required for both file-based import and bulk export.

Import and Export Data

In Oracle Sales Cloud, you can import and export data using two processes: file-based import and bulk export.

File-based import supports the import of data from an external text or xml file to interface tables and then from interface tables to target application tables.

> **Note:** File-based import bypasses any Groovy validation and trigger logic on an object. For example, object workflows are not triggered by an import.

Use bulk export to extract large volumes of data from Sales Cloud objects, both as extracts of a full set of records for an object as well as incremental extracts. Comma or tab-delimited files are created with the extracted data, which are available to users as attachments to the batch records that have been executed.

Enabling Import and Export for Custom Objects

The changes you make using Application Composer do not create the artifacts required by these import and export processes.

Accordingly, after completing your object model extensions, generate the required artifacts to register your extensions and make them available for importing and exporting.
Note: The creation of import and export artifacts occurs only in the Oracle Metadata Services mainline metadata, and is not supported within a sandbox.

To enable the import and export of custom object data:

1. Confirm that you're not in a sandbox.
2. On the main Overview page in Application Composer, select the Import and Export link in the Common Setup pane, or in the local area of the main Overview page.
3. On the Import and Export page, click the Generate button.

After you enable your object model extensions for importing and exporting, you can then schedule your file-based import and bulk export processes. For instructions and examples, see the Importing and Exporting Custom Objects chapter in the Oracle Sales Cloud Understanding File-Based Data Import and Export guide.

Related Topics
- File-Based Data Import and Export of Custom Objects: Explained
- Bulk Export: Overview

FAQs for Using Application Composer

What job role must I have to create my own objects in Application Composer?

Users with any one of the three following job roles can create their own objects and use all other Application Composer functions:

- Customer Relationship Management Application Administrator.
- Application Implementation Consultant.
- Master Data Management Application Administrator.

Oracle recommends provisioning the user with the Customer Relationship Management Application Administrator job role (for performing the configurations) and the Custom Objects Administration job role and Sales Administrator job role (for testing the configurations in the Oracle Sales Cloud UI).

What's the difference between fixed choice lists and dynamic choice lists?

A fixed choice list and a dynamic choice list are similar in that the ultimate goal of both types of choice lists is to generate a field with a list of values at run time. However, the list of values for a fixed choice list is derived from an FND lookup type. The list of values for a dynamic choice list is derived from an existing object’s actual data.
What's the difference between Page Composer and Application Composer?

Page Composer is a web-based tool you can use to modify user interface (UI) pages and components for all products designated for use with Page Composer. Page Composer uses two different modes of Design View. The first mode, Design View: Standard mode, is selected by default in all Sales Cloud pages when opening a page with Page Composer with the Design button selected. The second mode, Design View: Direct Selection mode, is activated when you click the Select tab for the UI page you want to modify. In Sales Cloud, Direct Selection mode is available when you modify pages, but not when you personalize a dashboard page. With the Design View: Direct Selection mode, you can select and edit UI elements such as form fields and table columns. In Direct Selection mode, the UI components that you can select become apparent when you move your cursor over them. The UI components that you can select are highlighted and can be edited.

The following table describes how you can use each mode of Page Composer to modify dashboard pages and other select pages (such as the Partner Public Profile page, Partner Landing page, Partner Registration, Customer Snapshot, and Customer Overview - Analysis tab), and modify transactional pages (all other non-dashboard pages).

<table>
<thead>
<tr>
<th>Use Cases</th>
<th>Design View - Standard mode</th>
<th>Design View - Direct Selection mode</th>
<th>Page Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add content (Business Intelligence reports, Sales Cloud portlets such as Calendar)</td>
<td>Yes</td>
<td>No</td>
<td>Dashboard and other select pages</td>
</tr>
<tr>
<td>Delete region</td>
<td>Yes</td>
<td>No</td>
<td>Dashboard and other select pages</td>
</tr>
<tr>
<td>Move region</td>
<td>Yes</td>
<td>No</td>
<td>Dashboard and other select pages</td>
</tr>
<tr>
<td>Change page layout (for example, change a two column layout to three column layout)</td>
<td>Yes</td>
<td>No</td>
<td>Dashboard and other select pages</td>
</tr>
<tr>
<td>Default region state (open or close)</td>
<td>Yes</td>
<td>No</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Manage save queries (create and edit)</td>
<td>Yes</td>
<td>No</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Hide or show field</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Change field label</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Make field required or not</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
</tbody>
</table>
Use Cases | Design View - Standard mode | Design View - Direct Selection mode | Page Type
--- | --- | --- | ---
Make field read-only or updatable | No | Yes | Transactional pages (all non-dashboard pages)
Reorder fields in a Form | No | Yes | Transactional pages (all non-dashboard pages)
Reorder table columns | Yes | Yes | Transactional pages (all non-dashboard pages)
Hide or show table columns | Yes | Yes | Transactional pages (all non-dashboard pages)
Set table column width with the mouse | Yes | No | Transactional pages (all non-dashboard pages)
Set table column width and minimum width in percent or pixels | No | Yes | Transactional pages (all non-dashboard pages)
Sort column or not | No | Yes | Transactional pages (all non-dashboard pages)

Application Composer also lets you make UI changes at run time. However, the types of UI changes that you can make using Application Composer are quite different. Specifically, your primary focus when using Application Composer is to make actual object model changes. For example, you can create a new business object and related fields, and then create new application pages where that object and its fields are exposed to users.

The following table describes some of the primary differences between Page Composer and Application Composer. For example, using Application Composer, you cannot access the Resource Catalog to add new content to a page.

<table>
<thead>
<tr>
<th>Task</th>
<th>Available in Page Composer (site, job role, external or internal level)?</th>
<th>Available in Application Composer (site level only)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make object model extensions and expose your changes by creating or modifying work area pages</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reorder subtabs</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Modify dashboard pages</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add content from the Resource Catalog</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Simple field changes (show, hide, make read only, make required)</td>
<td>Yes (WYSIWYG - what you see is what you get)</td>
<td>Yes (non-WYSIWYG)</td>
</tr>
<tr>
<td>View results of changes immediately</td>
<td>Yes, in the Page Composer design interface</td>
<td>Yes, in the Sales Cloud application that you are making changes to</td>
</tr>
</tbody>
</table>
5 Extending Simplified Pages

Overview

Modify Oracle Sales Cloud simplified pages and create page layouts that some or all users can see, depending on the conditions you set. Simplified pages are available for custom objects as well as most standard objects.

This chapter covers:

- Which simplified pages are extensible, what you can do, and how to modify simplified pages using Application Composer
- How to modify the springboard and the look and feel of the simplified pages
- Adding, enabling, or disabling drill down fields
- What dynamic page layouts are and how you use them

Tip: You can also modify simplified pages using Page Composer. Refer to the Page Composer chapter in this guide to learn more.

To modify simplified pages using Application Composer, select an object that offers a set of simplified pages, such as Opportunity, then select the Pages node. Select the Simplified Pages tab to access that object’s configuration pages, where you can modify the simplified page regions that are extensible.

Related Topics

- Extending Pages for Accounts: Explained
- Extending Pages for Leads: Explained
- Extending Pages for Opportunities: Explained

Modifying Oracle Sales Cloud Simplified Pages: Explained

After creating custom objects and updating standard objects in Application Composer, you must create or update user interface pages so your changes are visible to your end users. To work with simplified pages, select the Simplified Pages tab after expanding the object’s Pages node. This tab lets you create new pages for a custom object and modify existing pages for standard objects. You can modify these simplified pages (if they are extensible) using Application Composer and also Page Composer. For example, you can show or hide fields, rearrange fields, and add custom fields to simplified pages.

In this topic, you will learn which objects have simplified pages that are extensible, and how to modify simplified pages using Application Composer.

Extensible Content

You can create a set of simplified pages for all custom objects. Many standard objects offer a set of simplified pages, most of which you can modify.
Objects whose simplified pages are extensible have a tab called Simplified Pages when the Pages node is selected. You can modify the pages listed under this Simplified Pages tab.

For example, the following standard objects have a set of simplified pages which you can modify:

- Account
- Activity
- Asset
- Contact
- Customer Contact Profile
- Forecast Territory Details
- Household
- Lead
- Note
- Opportunity
- Partner
- Program Enrollments
- Relationship
- Resource
- Sales Campaign
- Sales Lead

Modify Content

Use either Application Composer or Page Composer to modify a set of simplified pages, if they are extensible.

Use Application Composer to:

- Hide or show custom fields

  Tip: When creating custom fields, set a maximum width of 15-20 characters for optimum display on Oracle Sales Cloud simplified pages.

- Hide or show standard fields
- Extend form regions
- Extend table regions
- Reorder fields (tables and pages)
- Change field labels
- Add subtabs

Use Page Composer to:

- Make application changes that are role-based
- Hide or show fields
- Change field labels
Modifying Simplified Pages Using Application Composer

To modify simplified pages using Application Composer:

1. On the main Overview page in Application Composer, select an object that offers a set of simplified pages, such as Opportunity in the object tree.
2. Select the **Pages** node.
3. Select the Simplified Pages tab.
4. Use the links on the tab to navigate to the object’s configuration pages, where you can modify the simplified pages that are available for the selected object.

   For example, show or hide fields, rearrange fields, and add custom fields.

   _Tip:_ Changes that you make to a simplified page aren’t automatically replicated on the object’s corresponding desktop page.

You can also modify simplified pages using Page Composer. Refer to the Page Composer chapter in this guide to learn more.

Modifying Simplified Pages Using Page Composer: Procedure

On a simplified page, you can modify user interface (UI) components by updating their properties, for example to change field labels, hide the component, or make a check box required.

When you start modifying simplified pages, by default, you can use the Design view. In the Design view, you can add content and make layout changes only in some pages. For other pages, you must use the Source view to make such changes.

_Note:_ Any changes you make apply:

- Only to the page you’re on.
- To all or specific groups of users, depending on the context layer you select before making changes.

Prerequisites

Activate a sandbox.

Modifying a UI Component

To update component properties:

1. Click your user image or name in the global header and select **Customize Pages**.
2. Select a context layer, for example to make changes only for users with a specific job role.
When you modify a UI component for a specific job role, that job role must be assigned to you for you to test the application change in the sandbox. Your security administrator can either assign the job role to you directly, or make the job role self-requestable for you to add it yourself from the resource directory.

3. By default, you start in the Design view, which lets you navigate to the component you want to modify. You can tell you’re in this view when the Design button on top of the page is highlighted. To use the Source view, you must select Source from the View menu. This menu isn’t displayed by default. To display the View menu, and then select Source, set the Source View for Page Composer Enabled profile option (FND_PAGE_COMPOSER_SOURCE_VIEW) to Yes.

4. When you have found your UI component, click the Select button on top of the page.

5. Hover over the UI component until a border appears around the component, and click.


7. Update the component’s properties to make the application change you want. Each component has its own set of properties, which may include some of the properties in this table. In Design view, you get the main properties only; but in Source view, you get all properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Text used by screen readers, for information in addition to what is provided in the Short Desc property.</td>
</tr>
<tr>
<td>Label</td>
<td>Display text for the component, for example the field prompt or the single prompt for a group of check boxes.</td>
</tr>
<tr>
<td>Read only</td>
<td>Whether users can edit the component, for example if a check box can be selected or not.</td>
</tr>
<tr>
<td>Rendered</td>
<td>Whether the component is visible or hidden to users on the page.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether users must enter something for the component before saving the page.</td>
</tr>
<tr>
<td>Short Desc</td>
<td>Text that appears when users hover or focus on the component, for example hover over a field label or click in the text box.</td>
</tr>
<tr>
<td>Show Component</td>
<td>Whether the component is visible or hidden to users.</td>
</tr>
<tr>
<td>Show Required</td>
<td>Whether an asterisk is displayed to indicate that the component is required.</td>
</tr>
</tbody>
</table>

8. To modify more components, click Add Content to return to the Design mode and repeat steps 4 to 7.

9. Click Close to save.

If available, click Save and Label instead to also label your changes so that you can later revert to the application changes you’re saving. Labels are saved with a prefix of composer_. For example, if you enter myLabel, then the label is composer_myLabel.

Related Topics
- Working With Context Layers: Examples
Creating a Set of Simplified Pages for Custom Objects: Explained

When you create a custom object, you can optionally create a set of simplified pages with a click of a button. When you click that button, Application Composer automatically creates default custom layouts for the object’s landing page (list page), creation page, and details page (edit page). A default custom layout for the search and select dialog is also created. Working with these page layouts is exactly the same as working with the page layouts for standard objects.

Creating Simplified Pages for Custom Objects

To create a set of simplified pages for a custom object:

1. Ensure that you’re working in an active sandbox.
2. In Application Composer, navigate to the object that you want to create page layouts for.
3. Expand the object in the object tree, and select the Pages node.
4. Select the Simplified Pages tab.
5. Click the Create Default Pages button.

Application Composer automatically creates a set of simplified pages for the custom object:

- A default custom layout for the landing page
- A default custom layout for the creation page
- A default custom layout for the details page
- A default custom layout for the search and select dialog, also known as a picker

6. Working with the default custom layouts created for custom objects is the same as working with the standard layouts for standard objects.
Selecting the Display Icon for Custom Objects

The simplified pages for objects are associated with a display icon. Application Composer creates custom objects with a default icon, but you can change it. The icon you select will display to your end users in a variety of locations, such as on the Navigator, subtabs, mobile pages, and the springboard strip on simplified pages.

Select the display icon for objects on the object's Overview page. See: "Defining Objects" for instructions.

Viewing Custom Object Simplified Pages at Run Time

After configuring the simplified pages for your custom object, you can test the run time pages by clicking the Home icon. Depending on which application you created your custom object in, you might have to wait a few minutes for the new display icon to appear on the Home page.

Tip: Navigate quickly and easily between Oracle Sales Cloud run time pages and Application Composer design time pages using the Favorites and Recent Items menu.

Related Topics

- Defining Objects: Explained
Configuring a Search and Select Dialog Box: Explained

Read this topic to learn about how to configure custom object Search and Select dialog boxes that users launch from simplified pages. A Search and Select dialog box, also known as a picker, lets your end users search for and select object records when assigning one record to another, such as a salesperson to an account or a solution to a service request. Search and Select dialog boxes are automatically provided for standard objects, and they’re not extensible. However, you must configure the Search and Select dialog boxes for the custom objects you create.

Search and Select Dialog Boxes

A Search and Select dialog lets your end users search for and select object records at runtime. For example, your users might need to assign solutions to service requests, but what if your company’s knowledge base includes over 1,000 solutions? Your users don’t want to scroll down a list of 1,000 solution records. In such cases, the Search and Select dialog box lets users provide search criteria in advance (for example, all solutions relating to a particular product) to more easily find what they’re looking for.

You can launch Search and Select dialog boxes from two areas:

- From a dynamic choice list field, commonly referred to as a list of values.
- From a subtab.

Configuring a Search and Select Dialog Box for Custom Objects

When you create a set of simplified pages for a custom object, Application Composer automatically creates a Search and Select dialog box, which you must then configure. Custom objects don’t have Search and Select dialog boxes until you create and configure them. When you create the dialog boxes, they are reused wherever you want to associate that custom object with any other object.

🔗 Note: Standard objects already have their own Search and Select dialog boxes, which are not extensible.

To configure a Search and Select dialog box for a custom object:

1. On the Simplified Pages tab for a custom object, click **Create Default Pages** if you haven’t yet created the set of simplified pages for a custom object.

   See "Creating a Set of Simplified Pages for Custom Objects: Explained."

2. On the Simplified Pages tab for the object, navigate to the Reusable Regions region, and edit the **Default custom layout**. You could also duplicate the Default custom layout to create and edit a new custom layout.

3. On the Edit Search and Select Dialog Layout page, configure both the search and table regions of the Search and Select dialog.

Using Custom Object Search and Select Dialog Boxes: Examples

After you create and configure a Search and Select dialog for a custom object, you will then automatically use that dialog whenever you want your users to associate that custom object with any other object. Let’s look at some examples.
For example, maybe your users want to assign a solution to a service request. In this case, they will assign a solution to a service request using a dynamic choice list field.

1. Create a custom object for the Solution.
   a. Create the set of default simplified pages for the object.
   b. Configure a Search and Select dialog box for the Solution object.

2. Create a custom object for the Service Request object.
   a. Create a dynamic choice list field, Solution, that is populated with records from the Solution object. This field will automatically use the Search and Select dialog that you configured in step 1.a.
      When you create the dynamic choice list field, Application Composer creates a one-to-many relationship between the Solution and Service Request objects. In other words, one solution can be associated with multiple service requests, but a service request can have only one solution.
   b. Add the new Solution field to the Create Service Request and Edit Service Request pages.

At runtime, your users will use the Solution’s Search and Select dialog box, available from the Solution dynamic choice list field, to search for and select a solution to assign to a service request record.

Tip: You could optionally add a Service Requests related object subtab to the Edit Solution page, if you want users to view all service requests associated with a solution. To do this task, however, you must create a Search and Select dialog for the Service Request object.

In this next example, let’s assign solution records to the company employee who authored the solutions. To accomplish this, you would use the same Search and Select dialog box that you configured for the Solution object on a subtab.

1. Create a one-to-many relationship between the Resource and Solution objects, using the Relationships page, available under the Common Setup pane in Application Composer.
   In this case, one resource can author multiple solutions, but a solution can have only one author.

2. On the Edit Resource page, create a related object subtab called Solutions that is populated with records from the Solution object.

At runtime, your users can search for and select one or more solutions to assign to a resource. They can also create a new solution record to assign to the resource, right from the Search and Select: Solutions dialog.

Related Topics
- Defining Pages: Explained

Working with Dynamic Page Layouts

Dynamic Page Layouts: Explained

Using Application Composer, you can present the same Sales Cloud page to your users, but display different page layouts depending on the conditions you define. For example, a sales executive might see certain privileged fields on an opportunity record, which other sales team members can’t see. Similarly, an open opportunity might have certain fields related to it being in progress, which won’t display on a closed opportunity.
Read this topic to learn about using dynamic page layouts:

- Where can you use dynamic page layouts?
- Examples of dynamically controlling the display of page layouts, based on:
  - Role/privileges of the user
  - Groovy expression
  - Type of record

To learn how to create page layouts, see "Working With Page Layouts: Explained."

**Where Can You Use Dynamic Page Layouts?**

The page layouts you create are restricted to simplified pages only. Page layouts are not available for desktop pages. You can create page layouts for these types of simplified pages:

- Landing page (list page)
- Creation page
- Details page (edit page)
- Search and select dialog

Create page layouts for certain standard objects, such as the following objects:

- Account
- Activity
- Contact
- Forecast Territory Details
- Household
- Note
- Opportunity
- Partner
- Relationship
- Sales Campaign
- Sales Lead

For a complete list of the standard objects that have simplified pages, see "Modifying Oracle Sales Cloud Simplified Pages: Explained."

You can also create page layouts for custom objects.

**Controlling When Page Layouts Are Displayed**

When you create a page layout, you set one or more conditions to control when that layout will be displayed. The conditions you can attach to a layout include:

- Type of record
  - Not supported for landing page (list page) custom layouts.
- Role
• Advanced expression

Not supported for landing page (list page) custom layouts.

The record type and role conditions are convenient, declarative ways of attaching conditions to a layout. Supply an expression to control the display of a layout, only if the record type and role conditions don’t meet your needs.

Examples of layout conditions include:

• Type of record
  
  o Display a qualification subtab with a questionnaire, when an opportunity is in the Qualification sales stage. But, display a close plan subtab with a checklist capturing critical data, when an opportunity is in the Close sales stage.
  
  o Only display the Closed Reason field on an opportunity, when the opportunity is closed.
  
  o Large organizations have multiple divisions, and each division might have different business process requirements. For example, Division 1 allows sales representatives to create orders from an opportunity, while Division 2 does not allow this.
  
  o Display different page layouts depending on the product category. For example, display different fields if the product category is a physical item, or if it’s a service pack.
  
  o Display different page layouts depending on type of activity, such as a telephone call, task, or appointment.

• Role
  
  o A sales manager might see fields related to approving an opportunity, whereas the sales representative would not see those fields.
  
  o A channel manager typically needs to see a different opportunity layout from a sales representative. For example, the channel manager might see a region on a page with fields related to the partner, program, and partner registration.
  
  o Some opportunity fields might apply only to field sales representatives, some fields to inside sales representatives, and some fields to follow-up sales representatives.

• Expression
  
  o Do not allow users to add a revenue item or a product to an opportunity, after a quote has been generated and approved, or while an opportunity is in approval.
  
  o Control the display of page layouts based on the user’s location, language, or device.
  
  o Control the display of page layouts based on the values of other choice list fields, not just the Record Type field.
  
  o Write an expression to combine multiple conditions.

Field Groups: Explained

A page layout is a page design that you create by selecting which fields to display on a page. When you design a page layout, you can use field groups to organize your pages and make them look more readable. A field group lets you group fields into collapsible regions, each with its own header that you can modify.

Which Fields Are You Grouping?

The fields that you can select for a field group are attributes of the top-level object that you’re creating the page layout for, such as the opportunity object.
Why Use Field Groups?

Field groups are useful for managing your page layouts:

- Group related fields so they always appear together on a page.

  Perhaps you want a group of fields, such as Home Ownership and Purchase Date of Home, to always appear together. Create a field group and add those two fields to the group.

- Group secondary fields in a region that your end users can optionally expand, if they need to.

  Maybe some fields on a page are useful, but not critical for your end users. Define the field group so that the region is always collapsed by default at run time.

- Manage page layouts with fewer clicks.

  Once you add a group of fields into a field group, you can easily move that group of fields up or down the page layout, with a single click.

  Multiple field groups always appear together at run time within a larger field group "container". When designing a page layout, you can move a field group up or down, but only within this larger container.

  In most cases, field groups appear at run time as regions right below the page’s top summary region.

Field Group Validation

Application Composer validates the contents of field groups: once you add a field to a group, you can’t add the same field to another group. This validation applies only across the field groups created for one page type (creation page or details page).

Tip: Although you can’t add the same field to multiple field groups, you can easily move a field between groups. This makes it easy to manage fields within groups, if you later change your mind about field placement.

Creating a Field Group

You create field groups as part of either a creation page layout, or a details page layout.

To learn how to create field groups, see Working With Page Layouts: Explained.

Working With Page Layouts: Explained

A page layout is a design of a page which you create, and then attach conditions to. Using conditions, you can present the same Sales Cloud page differently to various users. For example, a sales manager can see one version of the page, while the sales representative sees another version. Page layouts are available only for simplified pages.

Read this topic to learn how to create page layouts. This topic explains how to:

- Work with page layouts for standard objects
- Create a set of page layouts for custom objects
- Understand the difference between standard vs. custom layouts
- Duplicate and edit page layouts
- Add field groups
- Add conditions
Working with Page Layouts for Standard Objects

Standard objects that have simplified pages are delivered with default page layouts, called standard layouts. Standard layouts are the pristine model layouts that you can’t edit. However, you can duplicate the standard layout to create a new custom layout. You can edit custom layouts and add display conditions to them. When a custom layout is no longer of use, you can inactivate, or deprecate, that layout.

You can:

• Duplicate page layouts

To create a new page layout, duplicate an existing layout and then make your edits.

All supported objects are delivered with a standard layout for their simplified user interface pages. Duplicate the standard layout to create custom layouts, which you can edit.

The first custom layout for a page type is automatically named the default custom layout, but you can change the name.

• Edit custom layouts

You can edit only custom layouts. Custom layouts are duplicated from an existing layout.

• Inactivate, or deprecate, custom layouts

You can’t delete page layouts, but you can inactivate custom layouts by deselecting the Active check box for a page layout on the Simplified Pages tab.

You can’t inactivate the standard layouts that are automatically delivered for an object.

Working with Page Layouts for Custom Objects

Working with page layouts for custom objects is exactly the same as working with page layouts for standard objects. The only difference is that after you create a custom object, you must manually create its set of custom layouts before you can start to work with them. Custom objects do not have a set of standard layouts.

Create a set of simplified page layouts for a custom object with a click of a button. Clicking that button tells Application Composer to automatically create the following:

• Page layouts for the object’s set of user interface pages, such as the creation and details pages.

To learn how to create a set of simplified pages for a custom object, see Creating a Set of Simplified Pages for Custom Objects: Explained.

• A Search and Select dialog, which you can configure by clicking the Edit Picker link.

To learn how to configure the Search and Select dialog for custom objects, see Configuring a Search and Select Dialog: Explained.

Standard vs. Custom Layouts

Standard layouts are the pristine model layouts that you can’t edit. Custom layouts are copies of standard layouts that you make, which you can edit.

Standard layouts exist to make your upgrades seamless. When you upgrade to a new release of Oracle Sales Cloud, Oracle upgrades only the standard layouts for each object. Your custom layouts aren’t touched. This makes it easy for your users to continue working immediately after an upgrade. In the meantime, you can take your time to review the changes that happened to standard layouts as part of an upgrade, and manually incorporate those changes as and when needed.
After an upgrade, you can easily review the newly upgraded standard layouts by deactivating all existing custom layouts for a page type, such as the creation page. Then, log in as a user to view the standard layout at run time. Observe the changes for the creation page and, if desired, navigate back to Application Composer to incorporate those changes into your creation page custom layouts and reactivate them.

If an object has one or more custom layouts for some page types, but not for others, then Oracle considers the whole set of pages for that object to be modified. As part of the upgrade, Oracle:

1. Doesn’t touch the custom layouts that already exist, as usual.
2. Creates custom layouts for those pages that don’t have any custom layouts. These new custom layouts preserve what users experienced before the upgrade.
3. Upgrades all standard layouts, as usual.

Since standard layouts are model layouts that you can’t edit, this means that you can’t make changes to the page using Page Composer if the standard layout displays at run time. However, customers can still personalize the page.

**Editing Page Layouts**

When you edit a custom layout, you can interact with items inside regions, or interact with the regions themselves. You can:

- Add fields, actions, links, buttons, and subtabs.
- Hide and show, reorder, and relabel regions, including subtabs.

To edit a custom layout:

1. In Application Composer, navigate to the object that you want to create custom layouts for.
2. Expand the object in the object tree, and select the Pages node.
3. Select the Simplified Pages tab.
4. Find the type of page that you want to modify, such as a creation page layout or a details page layout. Click the **Duplicate Layout** icon to duplicate and edit an existing layout.
5. Next, design the page layout. Depending on how the page is designed, you can add fields, actions, links, and buttons.

You can also hide and show, reorder, and relabel regions.
If the page layout is for a details page, then you can also add and reorder subtabs, and add the Attachments field.

**Creation Layout: Default custom layout**

* Layout Name: Default custom layout

Create Opportunity: Hide

- Name
- Account
- Primary Contact
- Win Probability

**Adding Field Groups to a Page Layout**

When selecting the fields to include in a page layout, you can select the fields one by one, or add multiple fields as a single unit, called a field group.
To create a field group:

1. When editing a custom layout, click the New icon in the Field Groups region.

2. On the Create Field Group: Configure Field Group Details page:
   a. Enter the name of the field group. At runtime, the name is displayed as the name of this collapsible region.
   b. Indicate if the region is automatically expanded, or collapsed by default.
   c. You can also set the position of the field group in relation to other field groups, if other field groups already exist.
3. Next, add fields to the group.

Tip: You can multi-select and double click fields in the Available Fields list to move them to the Selected Fields list.
4. If additional custom layouts exist, then you can click Next to add this field group to other custom layouts.

For example, in the screenshot above, we are creating a field group as part of a layout for a creation page. But, you could also add this field group to the Sales Manager layout, if it existed.
5. Click **Save**. Your new field group now appears in the Field Groups region.

6. Once your field groups are defined, you can optionally hide field groups, or reorder field groups within the larger field groups “container” on the custom layout.
7. You can also move fields between groups, since a field can appear within only one field group for a particular custom layout.

8. When you are finished making changes to your custom layout, click **Done**.

**Assigning Conditions to Page Layouts**

After you edit the custom layout, you can assign one or more conditions that control when the layout is displayed. You assign conditions to custom layouts when viewing them on the Simplified Pages tab. You can’t assign conditions to the standard layout.

**Note:** The layouts you create are displayed in a table, and the order of layouts in each table is significant. At run time, Application Composer evaluates the condition or conditions specified in each layout, starting with the first layout listed in the table. The first layout that matches all Type, Role, and Expression conditions is selected for display at run time. The standard layout is always the last layout in the table, and it can’t be deleted or inactivated.

Assign one or more of these conditions to a custom layout.

- **Type**
  - a. Select the custom layout you want to add conditions to.
  - b. Select a record type field value, if a record type field has been created for the object. At run time, if the value is selected, then this custom layout displays.
The default value for this condition is ANY, so if you do not specify a Type condition for a layout, then Application Composer views this condition as satisfied when evaluating a layout for display at run time.

You can’t assign this condition to landing page (list page) custom layouts.

- **Role**

  a. Select the custom layout you want to add conditions to.
  b. Select the role; this is the audience that can view this page layout. For example, perhaps only the sales representative can see this custom layout at run time. Custom roles, which are copies of the predefined roles that Oracle provides for all customers, are displayed by default for you to select. However, you can optionally choose to display predefined roles, as well.

  The default value for this condition is ANY, so if you do not specify a Role condition for a layout, then Application Composer views this condition as satisfied when evaluating a layout for display at run time.

- **Advanced Expression**

  a. Select the custom layout you want to add conditions to.
  b. Click the Calculator icon.
  c. Enter a Groovy expression that controls when this custom layout is displayed.

You can’t assign this condition to landing page (list page) custom layouts.

**Tip:** In general, it’s best to keep a single custom layout, for each page type, condition-free. This way, if no conditions apply at run time, then at least your users see a generic custom layout. Otherwise, if no conditions apply at run time, then the standard layout displays.

### Controlling the Display of a Page Based on a Field Value: Explained

Using Application Composer, you can optionally present a different page layout to your users, depending on the conditions you define. One condition that you can set for a layout is based on the type of record, which your end users indicate by selecting a value from a field at run time. For example, if an opportunity is open, then certain fields might display. However, if the opportunity is closed, then other fields might display. Controlling the display of page layouts in this manner requires a special kind of choice list field, called a record type field.

### Controlling the Display of a Page Layout

To control the display of page layouts based on a choice list value, you must:

1. Create a record type field for an object.
2. Add the field to the desired simplified page where you want the field to appear, such as the object’s creation page or details page (edit page).
3. Assign each choice list value to a layout.

Landing page (list page) custom layouts do not support record type conditions.

At run time, when an end user selects a value from the field, the page display changes to match the simplified page layout that you associated with the field value.
Creating a Record Type Field

A record type field is a choice list field with a list of values that you specify.

Note: You can create only one record type field per object.

To create a record type field:

1. In Application Composer, navigate to the object that you want to create page layouts for.
2. Select the object itself to further expand the tree hierarchy.
3. Select the Fields node to navigate to the Fields page.
4. On the Fields page, click the Create a custom field icon on the Custom tab.
5. Select Record Type and click OK.
6. Enter basic field attributes, such as the field display name and whether or not the field is required and updatable.
7. Configure the list of values to display in the choice list. You can either select a predefined lookup type, or create a new one.
8. Configure which roles have which access to particular choice list values. In other words, you can restrict the list of values displayed at run time by role.

For example, perhaps the sales representative can see only selected choice list values, but the sales manager can see all the choice list values.

Custom roles, which are copies of the predefined roles that Oracle provides for all customers, are displayed by default. However, you can optionally choose to display predefined roles, as well.

Next, add the field to the desired simplified page layout, where you want the field to appear. This step is described in the next section.

Creating Page Layouts per Record Type

After creating the record type field, you must then add the field to the desired set of simplified pages, and then assign the choice list values to one or more page layouts. In this way, you control which layout displays at run time, depending on the choice list field value selected. You can’t assign this condition to landing page (list page) custom layouts.

To create a page layout for a record type:

1. In Application Composer, navigate to the object that you want to create page layouts for.
2. Expand the object in the object tree, and select the Pages node.
3. Select the Simplified Pages tab.
4. Find the type of page layout that you want to create for a record type. For example, maybe you want to create a page layout for display if an opportunity is open. In the Details Page Layouts region, select the standard layout and click the Duplicate Layout icon. Once you have a custom layout, you can make changes to it. You can’t add a record type condition to landing page (list page) custom layouts.

Note: The first custom layout that you create from a standard layout is called the default custom layout, but you can optionally change the name. Oracle recommends that you do not add conditions to the default custom layout. You can add record type conditions, and all other conditions, to subsequent custom layouts that you create.

5. Add the record type field to the selected layout.
6. After editing and saving the layout, select a choice list value under the Type column.

At run time, if an end user selects this value, then this layout displays.

Caution: Remember that during the creation of the record type field, you can also restrict the list of values by role. If you assign a Role condition to the layout as well, then confirm that both Role conditions are complementary.

Examples of Page Layouts per Record Type
Examples of page layouts that you might want to create for a record type field and its values are:

- Display a qualification subtab with a questionnaire, when an opportunity is in the Qualification sales stage. But, display a close plan subtab with a checklist capturing critical data, when an opportunity is in the Close sales stage.
- Only display the Closed Reason field on an opportunity, when the opportunity is closed.
- Large organizations have multiple divisions, and each division might have different business process requirements. For example, Division 1 allows sales representatives to create orders from an opportunity, while Division 2 does not allow this.
• Display different page layouts depending on the product category. For example, display different fields if the product category is a physical item, or if it’s a service pack.

• Display different page layouts depending on type of activity, such as a telephone call, task, or appointment.

**Related Topics**

• *Record Type Fields: Explained*

---

**Controlling the Display of a Page Based on a User's Role: Explained**

Using Application Composer, you can optionally present a different page layout to your users, depending on the conditions you define. One condition that you can set for a layout is based on the role of your end users. For example, a sales representative might see one particular layout, while the sales manager might see a different layout.

**Creating Page Layouts per Role**

Assign a role to one or more page layouts. In this way, you control which layout displays at run time, depending on the role of the user.

To create a page layout for a role:

1. In Application Composer, navigate to the object that you want to create page layouts for.
2. Expand the object in the object tree, and select the Pages node.
3. Select the Simplified Pages tab.
4. Find the type of page layout that you want to create for a role. For example, maybe you want to create a page layout for the sales manager. In the Creation Page Layouts region, select the standard layout and click the **Duplicate Layout** icon. Once you have a custom layout, you can make changes to it.

   The first custom layout that you create from a standard layout is called the default custom layout, but you can optionally change the name. Oracle recommends that you do not add conditions to the default custom layout. You can add role conditions, and all other conditions, to subsequent custom layouts that you create.

5. After editing and saving the custom layout, select a role under the Role column.

Custom roles, which are copies of the predefined roles that Oracle provides for all customers, are displayed by default. However, you can optionally choose to display predefined roles, as well.
Ideally, your roles should be stable before you assign roles to page layouts. If you make changes to roles after you create role-specific page layouts, then you will have to come back to these page layouts and update the assigned roles.

At run time, if the user has the specified role, then this page layout displays.

⚠️ Caution: If you assign a Type condition to the layout in addition to a Role condition, then confirm that the Role condition is complementary with any role assignments made at the record type field level.

Examples of Page Layouts per Role

Examples of page layouts that you might want to create for specific enterprise duty roles are:

- A sales manager might see fields related to approving an opportunity, whereas the sales representative would not see those fields.

- A channel manager typically needs to see a different opportunity layout from a sales representative. For example, the channel manager might see a region on a page with fields related to the partner, program, and partner registration.

- Some opportunity fields might apply only to field sales representatives, some fields to inside sales representatives, and some fields to follow-up sales representatives.
Controlling the Display of a Page Based on an Advanced Expression: Explained

Using Application Composer, you can optionally present a different page layout to your users, depending on the conditions you define. One condition that you can set for a layout is based on an expression you write using a Groovy script. For example, you might want to prevent users from updating a particular field based on the value of an opportunity. You write Groovy scripts using Application Composer’s expression builder.

Writing Expressions for Displaying Page Layouts

Write an expression for one or more page layouts. In this way, you control which layout displays at run time, depending on the expression. You can’t assign this condition to landing page (list page) custom layouts.

Tip: You can also control the display of pages using choice list values (using a record type field) and roles. The record type and role conditions are convenient, declarative ways of attaching conditions to a layout. Thus, supply an expression to control the display of a layout, only if the record type and role conditions don’t meet your needs.

To write an expression for a page layout:

1. In Application Composer, navigate to the object that you want to create page layouts for.
2. Expand the object in the object tree, and select the Pages node.
3. Select the Simplified Pages tab.
4. Find the type of page layout that you want to create. For example, maybe you want to write an expression that controls the display of a creation page. In the Creation Page Layouts region, select the standard layout and click the Duplicate Layout icon. Once you have a custom layout, you can make changes to it. You can’t add an expression condition to landing page (list page) custom layouts.

Note: The first custom layout that you create from a standard layout is called the default custom layout, but you can optionally change the name. Oracle recommends that you do not add conditions to the default custom layout. You can add an expression, and all other conditions, to subsequent custom layouts that you create.
5. After editing and saving the layout, under the Advanced Expression column, access the expression builder by clicking the calculator icon. Write an expression that describes the conditions required for this layout to display at run time.

If your script references one or more fields, then select those fields in the Depends On choice list, too. If those field values change at run time, then the expression is reevaluated and the page layout is refreshed if the new condition is met.

For example, let’s say you write this script:

```
PartyName == 'abc' || NoteTypeCode == 'GENERAL'
```

Your script references two fields. Thus, you must select those fields, Author and Type, in the Depends On choice list, which appears at the top of the expression builder.
Note that long text fields don’t work like other fields. If your script references a long text field, such as Note Text, then you must use the toString() operator in your script. For example:

```
If(NoteTxt?.toString() == 'abc')
```

Remember to select the Note Text field in the Depends On choice list, too.

- Your expression should return either a True or False value. At run time, Application Composer interprets a True value to mean that the condition was met.

For example, let’s say you want to display a specific page layout if the Win Probability for an opportunity is 95. In this case, your script could be:

```
if (WinProb==95)
{  
  return true;  
}
else 
{
  return false  
}
```

**Examples of Page Layouts That Display Based on Expressions**

Examples of page layouts that display based on expressions include:

- Do not allow users to add a revenue item or a product to an opportunity, after a quote has been generated and approved, or while an opportunity is in approval.
- Control the display of page layouts based on the user’s location, language, or device.
- Control the display of page layouts based on the values of other choice list fields, not just the Record Type field.
- Write an expression to combine multiple conditions.

**Configuring the Summary Table on a Landing Page: Worked Example**

When you configure an object’s landing page, you indicate the overall set of columns that are available for display in the summary table. You can indicate which columns should display by default, and which columns should be available for end users to manually display themselves as part of their own saved searches. After you configure the landing page, optionally assign a role condition to the page layout so that you can display a different set of columns on the page per role. For example, you may want your salespeople to see different landing page information than your partner managers.

At runtime, your end users can optionally create their own saved searches on the landing page summary table, choosing from the set of columns that you selected.

The following example shows you how to configure the summary table on a landing page.

**Configuring the Summary Table**

1. In Application Composer, navigate to the object whose landing page you want to configure.
2. Expand the object in the object tree, and select the Pages node. Confirm that you are on the Simplified Pages tab.

3. In the Landing Page Layouts region, select the standard layout and click the **Duplicate** icon. Once you have a custom layout, you can make changes to it. Or, edit an existing custom layout.

4. Navigate to the summary table region and click the **Edit** icon. Some standard objects might refer to the summary table as the overview table.

5. In the Configure Summary Table region, indicate which fields should display as columns in the landing page’s summary table.

6. For each field that you select, the **Display in Summary Table** check box is automatically selected, but you can deselect it. Deselecting the check box means that the column will not automatically display on the landing page at runtime. However, your end users can manually display that column when they create a saved search at runtime.

7. Click **Save and Close**.

8. After editing and saving the custom landing page layout, optionally select a role under the Role column to specify which users should see this landing page at runtime.

9. If you assigned a role condition, then sign in as that role and test your changes.

   **Note:** You must have the appropriate job role assigned to you to see the changes.

**Related Topics**

- Assigning Yourself Additional Job Roles Required for Testing

---

**Enabling or Disabling Drill Down Fields: Explained**

This topic covers how you enable or disable drill down fields on simplified pages. These drill down fields enable you to edit an object from the details page of another object.

You must familiarize yourself with the following topics before you configure drill downs:

- Extending Simplified Pages: Overview
- Object Relationships: Explained
- Dynamic Choice Lists: Explained
- Subtabs: Explained

**Overview**

The ability to drill down on a field is based on relationships that exist among the objects involved. These relationships could be either implicitly defined through a dynamic choice list type fields or joins, or explicitly defined by creating a relationship using Application Composer.

**Note:** You can create drill down fields for custom objects, and for standard objects whose task flow has been registered for extensibility. You cannot drill down to a common component or custom object.

You can configure drill down for the following:

- Details and Summary Pages
- Child or related object subtab and Context link subtab
Use Application Composer to add drill down fields to the desired pages using the Simplified Pages tab, and then view these pages on your laptop or tablet.

You can also drill down from one edit page to another in a hierarchical manner. For example, drill down from the Edit Leads page to an Edit Opportunity page, and then from the Edit Opportunity Page to the Edit Primary Contact page.

**Adding Drill Down Fields to Pages**

This section covers how you add drill down fields to pages. For information on how you create pages, see the related topic, Defining Pages: Explained.

When configuring a details page by moving fields from the Available Fields box to the Selected Fields box, the fields related to the object for which you are modifying the page appear in a table on the right. In the table, the drill down is enabled by default for related fields.

> **Note:** The table appears only if the selected fields are related.

This figure illustrates the table (on the right) where you can enable or disable drill down fields.

You can enable or disable drill downs using the Drill Down Enabled check box in the table. The check box is selected and grayed out for standard fields of type dynamic choice lists that are delivered by default.

For work area pages, you also specify the Drill Down Column. This column lists the field values as links.

**Adding Drill Down Fields to Subtabs**

This section covers how you add drill down fields on subtabs. For information on creating subtabs, see the related topic, Subtabs: Explained.

You can add drill down fields to the following types of subtabs:

- **Child or related object subtab**

  When creating a Child or related object subtab, in the Drill Down Field list, specify the field whose values should appear as drill down links in a column on the subtab.
This figure illustrates where you select and specify drill down column and fields.

The Selected Fields table (on the right) displays the fields available for drill down. Specify the fields that you want to display as drill down links on the subtab.

- **Context link subtab**

  When creating a Context link subtab, in the Drill Down Field list, specify the field whose values should appear as links in a column on the subtab.
This figure shows the page where you select and specify drill down column and fields.

You can also limit the fields you want to display on the subtab by specifying the filters under Search Criteria.

For more information on using filters, see the related topic, Subtabs: Explained.

**Related Topics**

- Object Relationships: Explained
- Dynamic Choice Lists: Explained
- Subtabs: Explained

**Specifying Drill-Down Fields for Custom Dynamic Choice List Fields: Example**

This example shows how you specify drill-down fields for custom dynamic choice list fields in simplified pages.
Specifying Drill-Down Fields

To specify a drill-down field for a custom dynamic choice list field:

1. Select a sandbox to work in and make it active.
2. Navigate to Application Composer.
3. Expand Standard Objects and click Sales Lead.
4. Click the Actions and Links link.
5. Click the Simplified Pages tab.
6. Edit the desired custom layout in the Details Page Layouts region.
7. Select the Summary tab, and click the edit icon.
8. Select a dynamic choice list item from the Available Fields column.

This figure shows the drill down status column for the selected dynamic choice list fields.

You have now specified the drill-down fields.

Creating and Adding Custom Links to Simplified Pages: Worked Example

This worked example illustrates how to create and add custom links to simplified pages.

This example covers:

1. Creating a custom link that launches the Google homepage.
2. Adding the custom link to the Leads page.
3. Verifying that the link appears on the UI.
Creating a Custom Link

In this step, you are creating a custom link called Test.

> **Note:** You must have a sandbox active before you begin your tasks.

1. Navigate to Application Composer.
2. Expand **Standard Objects**, and then expand **Sales Lead**.
3. Click **Actions and Links**.
4. In the Sales Lead: Actions and Links page, select **Actions > Create**.
5. Enter or select the following:

<table>
<thead>
<tr>
<th>Field or Region</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display Label</strong> field</td>
<td>Enter <strong>Test</strong>.</td>
</tr>
<tr>
<td></td>
<td>After you enter Test, the <strong>Name</strong> field automatically displays <strong>Test</strong>.</td>
</tr>
<tr>
<td><strong>Type</strong> field</td>
<td>Select <strong>Link</strong>.</td>
</tr>
<tr>
<td></td>
<td>After you select <strong>Link</strong>, the <strong>Source</strong> field automatically shows the <strong>URL</strong> option selected, and displays it as disabled.</td>
</tr>
<tr>
<td><strong>URL Definition</strong> region</td>
<td>Enter &quot;<a href="http://www.google.com">http://www.google.com</a>&quot; (along with double quotation marks).</td>
</tr>
</tbody>
</table>

6. In the script region, enter "http://www.google.com".
7. Click the **Validate** icon. A success message appears.
8. Click **Save**.

Next, you must add this new link called Test to the Leads page.

Adding the Link to the Leads Page

In this step, you add the Test link to the Leads page:

1. Expand **Standard Objects**, and then expand **Sales Lead**.
2. Click **Pages** under **Sales Lead**.
3. Select the Simplified Pages tab.
4. In the Details Page Layouts region, duplicate the standard layout to create a new layout to edit, or edit another existing layout.
5. In the Edit Simplified Details Page, select the vertical tab to which you want to add the custom link. In this example, select the Summary tab.
This figure shows the Summary tab and the Edit icon.

6. Click Edit icon in the Summary region. The Edit Lead page opens.

7. In the Available Fields column, locate the **Test - Link** field and move it to the Selected Fields column.

8. Click **Save and Close**.

**Testing the Link**

In this step, you verify the addition of the custom link.

To test whether the Test link appears on the UI:

1. Open the Leads page.
2. Edit a lead and note the Test link on the Edit Lead: <Name> Summary page.
This figure shows the Test link.

### Working with Subtabs

#### Adding and Hiding Subtabs Using Application Composer: Explained

Every top-level Oracle Sales Cloud object (such as an opportunity, account, or contact) has an edit page, also known as a details page. The details page is where users go to view record details, and make changes. You can use Application Composer to modify the details page. For example, you can show and hide fields, and you can also show and hide subtabs. Subtabs are useful because they display details that are related to the current record but derived from another record, or even from another source outside Oracle Sales Cloud. For example, when editing a trouble ticket record, your users might want to view the list of products that are reported on the trouble ticket. You create that subtab using Application Composer. Read this topic to learn more about subtabs in the simplified set of pages.

#### What's a Subtab?

A subtab lets you display details that are related to the current object but derived from another object, or from another source outside the current Oracle Sales Cloud application altogether. You do this by adding subtabs to the details page, and specifying the source of subtab data. At run time, your users can click each subtab to review data that is related in some way to the current object’s record.

A subtab displays data in a list. Users can click any record in the list to drill down to view more details about that record.

> **Note:** When viewing a subtab on an iPad or other iOS device, advise your users to scroll to the right directly on the subtab data rows. Scrolling to the right on the subtab area outside the data rows is not supported by iOS devices.

Let’s look at the subtabs that are available from the Edit Opportunity page. For example:

- **Contacts**
  This subtab lists records from the Opportunity Contact object, which is a child of the Opportunity object. Your users would click this subtab to review contacts that are related to the current opportunity record.
Tip: Child objects have a cascade delete relationship with parent objects. This means that if the parent record is deleted, then child records are also deleted.

- Opportunity Team

  This subtab lists records from the Opportunity Team Member object, which is a child of the Opportunity object. Your users would click this subtab to review team members that are related to the current opportunity record.

- Leads

  This subtab lists records from the Sales Lead object. This is a context link type of subtab, which displays data from any object. There is no cascade delete relationship between the Opportunity and Sales Lead objects. Your users would click this subtab to review leads that are related to the current opportunity record.
Where Do Subtabs Appear?
Subtabs display on an object’s details page. Every top-level Oracle Sales Cloud object has a details page, also known as the edit page, as part of its work area.

The details page is the page where users can view more details about an object. Depending on the security setup, users access the details page by clicking the Edit icon or by selecting the Edit menu item from the Actions menu on the summary table’s toolbar. Users can also access the details page by clicking the object record name itself in the summary table.

The details page can also display information related to the object record in subtabs. For example, the details page for an opportunity could include a subtab that lists customer contacts or previous orders.

Standard objects in Oracle Sales Cloud are delivered with a details page that has a set number of subtabs. You can add more if required.

For the custom objects you create, you can add subtabs once you create the work area for the object.

Adding Subtabs
Let’s say you want to add a new subtab to the Edit Opportunity page. In this example, let’s show the customer’s address. (For the sake of this example, let’s assume that you previously created a custom field, Customer Address (CustomerAddress_c), on the Opportunity object to capture this information.)

Use this procedure to add a new subtab to the Edit Opportunity page.

1. Navigate to Application Composer.
2. Under the Objects navigation tree, expand Standard Objects, then expand the Opportunity object.
3. Click the Pages node.
4. Click the Simplified Pages tab.
5. In the Details Page Layouts region, select a layout. You can select the standard layout and click the Duplicate Layout icon to duplicate it, or select another layout. Click the Edit icon.
6. On the Edit Simplified Details Page, scroll down through the Subtabs Region and, at the end of the subtabs, click the Add icon.

You can add one of five types of subtabs:

- Related object subtab
- Child object subtab
- Context link subtab
- Notes subtab
Mashup content subtab

These subtab types are described in “Subtab Types: Explained”. Read that topic to learn how to add each specific type of subtab.

Hiding Subtabs

Let’s say you want to hide the Appointments subtab on the Edit Opportunity page.

Use this procedure to hide subtabs.

1. Return to the Edit Simplified Details Page page for the Opportunity object.
2. Click the Configure Subtabs icon (green arrows).
3. In the Configure Subtabs dialog, in the Selected Subtabs list, double-click the Appointments subtab.
4. Click OK.
5. Click Done.
6. When you navigate to the Edit Opportunity page at run time, the Appointments subtab is no longer available.

Subtab Types: Explained

Subtabs display details that are related to the current record but derived from another object entirely, or even from another source outside Oracle Sales Cloud. Subtabs display on a top-level object’s details page. The details pages for standard objects are delivered with a set number of subtabs, but you can add more if required. You can add subtabs to the details pages for custom objects. Using Application Composer, you can create five types of subtabs.

Read this topic to learn about the different types of subtabs that you can create:

- Related object subtab
- Child object subtab
- Context link subtab
• Notes subtab
• Mashup content subtab

Related Object Subtabs

A related object subtab lists records from one object that is related to another object.

What does it mean when an object is related to another? A relationship is a foreign key association between two objects. Using Application Composer, you can create a one-to-many or many-to-many relationship between two objects. This is helpful because if a relationship exists, then you can expose the “many” object records on a subtab that is displayed on the “one” object’s details page. This is useful for your users.

For example, your users might want to associate an account with a list of service requests that have been logged for that account. To enable this, you must first create a one-to-many relationship between the account and service request objects. (An account can have multiple service requests associated to it, and a given service request can have one and only one account associated with it.) Then, you can add the Service Requests subtab to the Account details page. At run time, when your users review a particular account record, they can see all the service requests that have been logged for that account. And, depending on how you configure the subtab, they can also create new service requests, add existing service requests, or remove service requests from the subtab.

For custom related object subtabs that you create, your users can create or edit records inline in each related object subtab. But, you can also configure the subtab so that your users can create or edit records in a full-sized dialog window, in addition to inline editing. This is more usable if there are a large number of fields for users to fill out. This setup is described as follows.

To add a related object subtab to an existing details page:

1. Click the Create Subtab icon from either the Pages Overview page (for desktop pages) or from any details page layout (for simplified pages).
2. Select Related object.
3. On the Create Subtab page:
   a. Select the related object that is to be exposed on the subtab, and enter the subtab display label.
The list of related objects includes those objects that:

- Are already related to the current object, with any type of one-to-many or many-to-many relationship, either custom or standard.

  For example, if you previously created a one-to-many reference relationship, then you will see the "many" object in the data object list when creating the subtab for the "one" object’s details page.

- Don’t yet have a subtab.

After the subtab is created, you won’t be able to create a second subtab for that same "many" object again.

b. (Simplified UI only) Specify the drill down column. The drill down column is the field that you want your end users to click to drill down to more details about the related object record.

If you’re creating a subtab for a custom object, then the Drill Down Column field might not appear if you haven’t yet created the custom object’s simplified UI pages. To display the Drill Down Column field, you must first create the simplified UI pages for the custom object, sign out, and then sign back in.

c. Specify a display icon for the subtab.

d. Select which fields and links you want to display on the subtab summary table at run time.

   When selecting the fields for display on a related object subtab, join fields are not available for selection if the relationship is a many-to-many relationship.

e. Optionally hide the Create, Add, and Remove buttons that appear on the subtab at run time.

In a related object subtab, your users can use these buttons at run time to:

- Create a related object record and associate it to the current record at the same time.

  ✏️ Note: This button is available only for subtabs in simplified pages.

  - Add an existing related object record to the current record.
  - Remove a related object record from the subtab. This removes the association between the related object record and the current record.

f. (Simplified UI only) If more than one details page layout exists, then click **Next** to optionally select other details page layouts that will display this subtab.

g. (Simplified UI only) If you want your users to create or edit new records in a full-sized dialog window, in addition to inline editing, then complete these additional steps. This is more usable if there are a large number of fields for users to fill out.

   i. On the details page layout where the related object subtab appears, click the subtab that you want to configure.
   
   ii. Click the Edit pencil icon.
   
   iii. Click the Enable Separate Create and Edit Pages check box.
   
   iv. Click **Save and Close**.

Once completed, your users can create related object records in a separate full-sized dialog window. To edit existing records, they can click the record name to drill into a full-sized edit page.

Even after you enable this full-sized record creation and editing capability, your users can still opt to do their record creation and editing inline, if they choose to. Both options are now available to them.
This feature is available for related object subtabs based on custom top-level objects that are the "many" object in a one-to-many relationship with a standard Oracle Sales Cloud object. This feature is also available for child object subtabs based on custom child objects.

h. Click **Save and Close**.

**Child Object Subtabs**

A child object subtab lists records from a child object that are related to its parent.

A parent child relationship is a one-to-many relationship: one parent record can have many children records. When you create a child record, it’s created specifically in the context of its parent. The child object’s records are deleted if the parent object record is deleted. A child object doesn’t have its own work area, and exists only as a subtab on the parent object’s details page.

An example of a parent child relationship is the relationship between the Opportunity object, and its children, such as Opportunity Contact and Opportunity Partner. In this example, contacts and partners are created in the context of the parent opportunity record. At run time, when your users review a particular opportunity record, they can see all the contacts and partners that have been created for that opportunity. They can edit or add new contacts and partners directly inline, in each subtab.

For custom child object subtabs that you create, your users can create or edit records inline in each child object subtab. But, you can also configure the subtab so that your users can create or edit records in a full-sized dialog window, in addition to inline editing. This is more usable if there are a large number of fields for users to fill out. This setup is described below.

To add a child object subtab to an existing details page:

1. Click the Create Subtab icon from either the Pages Overview page (for desktop pages) or from the details page layout (for simplified pages).
2. Select **Child object**.
3. On the Create Subtab page:
   a. Select the child object that is to be exposed on the subtab, and enter the subtab display label.
   
   The list of child objects includes those objects that:
   
   • Are already related to the current object in a parent child relationship.
   • Don’t yet have a subtab.

   After the subtab is created, you won’t be able to create a second subtab for that same child object again.
   
   b. Specify a display icon for the subtab.
   
   c. Select which fields and links you want to display on the subtab summary table at run time.
   
   d. Optionally hide the Create, Edit, and Remove actions on the subtab at run time.

   In a child object subtab, you can hide these actions, or keep them available. If you keep these check boxes selected, then your users can:
   
   • Create a child object record inline.
   • Edit an existing child object record inline.
   • Delete a child object record inline.
   
   e. (Simplified UI only) If more than one details page layout exists, then click **Next** to optionally select other details page layouts that will display this subtab.
   
   f. Click **Save and Close**.
g. (Simplified UI only) If you want your users to create or edit new records in a full-sized dialog window, in addition to inline editing, then complete these additional steps. This is more usable if there are a large number of fields for users to fill out.

i. On the details page layout where the child object subtab appears, click the subtab that you want to configure.

ii. Click the Edit pencil icon.

iii. Click the Enable Separate Create and Edit Pages check box.

The Drill Down Column appears, but you can’t change it. It’s always the child object’s record name.

iv. Click **Save and Close**.

Once completed, your users can create child records in a separate full-sized dialog window. To edit existing child records, they can click the record name to drill into a full-sized edit page.

Even after you enable this full-sized record creation and editing capability, your users can still opt to do their record creation and editing inline, if they choose to. Both options are now available to them.

This feature is available for child object subtabs based on custom child objects. This feature is also available for related object subtabs based on custom top-level objects that are the "many" object in a one-to-many relationship with a standard Oracle Sales Cloud object.

Context Link Subtabs

A context link subtab displays a filtered list of records from any top-level object, where the filter is often based on the run time values from the current object. The object does not have to be related to the current object. Context link subtabs are read only.

To add a context link subtab to an existing details page:

1. Click the Create Subtab icon from either the Pages Overview page (for desktop pages) or from the details page layout (for simplified pages).
2. Select **Context link**.

3. On the Context Link subtab configuration page:
   
   **a.** Select the object that is to be exposed on the subtab, and choose the subtab display label.
   
   **b.** Optionally constrain the list of records displayed at run time using a set of search criteria for the selected object, whose run time values must match the current object record's run time values.

   Values can be literal values, or derived from the run time values in the current object record, or from the run time values in the current object's parent record.

   ✓ **Note:** If your search criteria includes a fixed choice list field, then you must specify the fixed choice list's run time value using the lookup code, not the lookup meaning.

   **c.** Select which fields you want to display on the subtab's read-only summary table at run time.

   You can configure fields for the main summary table which lists the child object records or related object records.

   **d.** Select which fields you want to display on the subtab's read-only detail form at run time.

   You can configure fields for the detail form that appears under the summary table.
e. (Simplified UI only) If more than one details page layout exists, then click **Next** to optionally select other details page layouts that will display this subtab.

f. Click **Save and Close**.

**Notes Common Component Subtabs**

A common component subtab adds a Notes subtab to show a list of notes related to a custom, top-level object. The Note object has a standard user interface (UI) that is shared across all Oracle Sales Cloud applications. To modify this UI, select the Note object, then select the Pages node on the object’s navigation tree to access the work area configuration pages.

At run time, end users can access the Note subtab and create a common component record that is tied to the object record. For example, a user can record a note on an service request record.

To add a Note subtab to an existing details page:

1. Click the Create Subtab icon from either the Pages Overview page (for desktop pages) or from the details page layout (for simplified pages).
2. Select **Notes**.
3. On the Common Component subtab configuration page, select **Note**.

**Mashup Content Subtabs**

A Mashup Content subtab exposes an external Web Site right on an object’s details page. The mashup content is a result of the expression that you define, which builds the intended URL. For example, on the Contact details page, perhaps you want to add a map using Bing maps that shows the location of the contact. The Bing Maps API expects the URL to be formatted in a certain manner. In this example, write an expression using the fields: Contact Address, Contact City, and Contact State. Then, pass the URL to the Bing Maps API.

**Prerequisites:**

Before you add a Mashup Content subtab:

- You must first register the web application that you want create the mashup for. See the Creating Mashups topic.
- You must also ensure that the registered mashup is active. See the Editing Mashups topic.

To add a Mashup Content subtab to an existing details page:

1. Click the Create Subtab icon from either the Pages Overview page (for desktop pages) or from the details page layout (for simplified pages).
2. Select **Mashup content**.

The Select Mashup Content page appears.

3. Select the mashup that you want to embed into your application.

4. Click **Insert**.

The Create Subtab page appears with the configured URL definition.
5. In the Display Label field, enter a name for the new subtab.
6. A default Display Icon appears. To change it, click **Change Icon**.
7. Based on the selected mashup type, do one of the following:
   - If the selected mashup is a parameter-based mashup, use the Add Parameters section to add values for each of the web application’s URL parameters configured while registering it. These parameters are appended to the web application’s URL as key-value pairs.
   - If the selected mashup is a groovy expression mashup, use the Edit Script section to edit the base URL and add to the URL definition of the registered web application.
8. Click **Next**.
   The Additional Layouts page appears.
9. (Simplified UI only) If more than one details page layout exists, then click **Next** to optionally select other details page layouts that will display this subtab.
10. Click **Save and Close**.
The embedded mashup appears at the bottom of the Subtabs Region page.

Note: Before end users can view the contents of your Mashup Content subtab at runtime, they must enable the settings in their selected browser to allow mixed content. Refer to the browser documentation for instructions on how to enable this option.

Related Topics
- Subtabs: Explained
- Adding Subtabs: Example

Creating and Reordering Subtabs in Simplified Details Pages: Worked Example

This example illustrates how to add and reorder subtabs that appear on the simplified details pages in Oracle Sales Cloud. In this example, let’s look at adding and reordering subtabs for the Account object.

Creating a Subtab
Let’s create a related object subtab for the Edit Account simplified page.

Note: Before you begin, you must be in an active sandbox session. Also, we will assume that you already created a custom child object for the Account object.
To create a subtab:

1. Navigate to the Account object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit, or edit another existing layout.
4. On the Details Layout page, click the + icon at the bottom left of the page. This icon appears at the bottom of all subtabs.
5. On the Create Subtab page, select Child object, and click Next.
6. On the Create Subtab: Child Object page, enter details for the child object whose records you want to capture on this custom subtab.
7. Click Save and Close.

Reordering a Subtab

Next, we can configure the custom subtab so that it appears closer to the top of other subtabs.

1. On the Details Layout page, click the Reorder Subtabs icon which appears as two green arrows at the top of the subtabs.
2. In the Configure Subtabs dialog, highlight your custom subtab and then use the up arrow to change the order of your subtab within the existing set of subtabs.
3. Click OK, and then click Done.

Adding a Related Object Subtab: Worked Example

This example illustrates how you add a related object subtab to a simplified details page.

This example covers the following:

1. Creating a related object subtab.
2. Verifying the addition of the subtab.

Creating a Subtab

In this step, you create a related object subtab using the Sales Lead object to display fields from the Opportunity object.

For more information about creating subtabs, see Subtabs: Explained.

To create a subtab:

1. Navigate to Application Composer.
2. Under the Standard Objects tree, expand the Sales Lead object.
3. Click the Pages node.
4. Select the Simplified Pages tab.
5. Under the Details Page Layouts region, select the page layout that you want to edit. In this example, select the standard layout and click the Duplicate Layout icon to make a copy of it to edit.
6. On the Edit Layout page, select the plus icon in the subtabs region.
7. On the Create Subtab page, select the Related object option.
8. Click Next.
9. Specify the values for the fields as follows:
### Extending Simplified Pages

<table>
<thead>
<tr>
<th>Field Label</th>
<th>Select or Enter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Object</td>
<td>Opportunity</td>
<td>The object that this subtab is based on. This subtab will display a list of opportunities that are related to a sales lead.</td>
</tr>
<tr>
<td>Display Label</td>
<td>Opportunity Name</td>
<td>Name of the subtab.</td>
</tr>
<tr>
<td>Drill Down Column</td>
<td>Opportunity Name</td>
<td>Your users will click this column to drill down to view more details about an opportunity record.</td>
</tr>
<tr>
<td>Display Icon</td>
<td>As applicable</td>
<td>This icon appears on the simplified page as the subtab icon.</td>
</tr>
<tr>
<td>Selected Fields</td>
<td>As applicable</td>
<td>The fields that you want to appear on the subtab.</td>
</tr>
</tbody>
</table>

10. **Save the Opportunity Name subtab.**

You have created a subtab to display opportunities on a subtab which appears on the Sales Lead details page. To view the new subtab, use the Navigator to view the Leads set of pages, and edit a sales lead record.

### Verifying the Addition of the Subtab

In this step, verify whether the Opportunity Name subtab has been added to the (Sales) Leads page.

1. Using the Navigator, select **Leads**.
2. Click a lead to open the **Edit Lead** page.

   The Opportunity Name subtab appears along the left side, with all the other subtabs.

You have now verified the addition of the subtab.

### Related Topics

- **Subtabs: Explained**

### Accessing Parent Data When Creating Subtab Records: Explained

When you add a subtab to any "parent" page, you can also configure the subtab to default parent data into new records. This means that when your users create a new subtab record, one or more fields can be automatically populated with data from the parent record. This capability is available when the subtab is based on either a standard or custom object, and it’s available for both related object subtabs and child object subtabs. This topic describes how you set this up.

### Relationships and Subtabs

In Oracle Sales Cloud, two objects can be related in either a one-to-many relationship or in a many-to-many relationship. Relationships are useful because they let you associate one object’s records with the records of another object. For example, maybe your users want to track the trouble tickets that get created for an account. You enable this association between records first by creating a relationship between the objects, then by creating the subtabs.
Accessing Parent Data from Subtabs

At run time, when a user creates a new record on a subtab, you can choose to have values from the parent record automatically default into a field on the subtab. You must manually enable this capability by writing Groovy to pull data from one field into another field.

To enable this capability, use the Groovy AfterCreate trigger to access parent object data. The trigger will copy parent data to the new record created on the parent object’s custom subtab. You must add this Groovy, as follows:

- If the subtab is based on a one-to-many relationship:
  - Add Groovy to the "many" object.
- If the subtab is based on a many-to-many relationship:
  - Add Groovy to the intersection object.

Let’s look at the following examples in the next section.

Groovy Example for a One-to-Many Relationship

You have a Vehicle subtab on the Account object. At run time, when your users create a new Vehicle record on this subtab, you want to populate the new Vehicle record with the Account’s city from the City field of the Account object.

To enable this, create an AfterCreate trigger on the Vehicle object with the following code:

```groovy
setAttribute('AccountCity_c', Account_AccToVehicle_Src?.PrimaryAddressCity);
```

- `AccountCity_c` is the target field on the Vehicle object that we want to populate with the value from the Account object’s City field.
- `Account` in `Account_AccToVehicle_Src?` indicates the source object from which to pull the data.
- `AccToVehicle` in `Account_AccToVehicle_Src?` is the name of the 1:M Account to Vehicle relationship.
- `PrimaryAddressCity` is the source data from the Account object that we will use to populate the `AccountCity_c` target field on the Vehicle object.

Groovy Example for a Many-to-Many Relationship

You have a Country Club Membership subtab on the Account object. At run time, when your users create a new Country Club Membership record on this subtab, you want to populate the new Country Club Membership record with the Account’s city from the City field of the Account object.

To enable this, create an AfterCreate trigger on the AccountCountryClub intersection object with the following code:

```groovy
setAttribute('AccountCity_c', Account_AccToCountryClub_Src?.PrimaryAddressCity);
```

- `AccountCity_c` is the target field on the Country Club Membership object that we want to populate with the value from the Account object’s City field.
- `Account` in `Account_AccToCountryClub_Src?` indicates the source object from which to pull the data.
- `AccToCountryClub` in `Account_AccToCountryClub_Src?` is the name of the M:M Account to Country Club Membership relationship.
- `PrimaryAddressCity` is the source data from the Account object that we will use to populate the `AccountCity_c` target field on the Country Club Membership object.
Subtabs Based on Self-Referencing Relationships

In the case of self-referencing object relationships, both 1:M and M:M, it’s not possible to access parent object data when creating records in custom subtabs. For example, it is possible to have the same object (such as Trouble Ticket) as both the source as well as the target of a 1:M or M:M relationship. However, when creating new records in a subtab for such a relationship, parent record data cannot be accessed.

Working with Mashups

Mashups: Explained

Using Application Composer, you can integrate an external web application with a Sales Cloud application by creating and embedding mashups.

What’s a Mashup

A mashup is an integration of a web application with Oracle Sales Cloud. By configuring a mashup, you can embed a web application in a Sales Cloud application page. This lets end users pull information from different sources to create a personalized application that meets their exact requirements.

For example: Assume that your company uses a third-party billing system and the sales agent needs the ability to view and modify the customer’s billing information. By creating a mashup, you can embed the third-party billing system within a subtab of your Sales Cloud application. The sales agents can quickly and easily view the customer’s billing information. If actions such as update and refund are available through the third-party page, sales agents can execute them from within the subtab.

To create a mashup, you register a web application in Application Composer and then embed it directly into a Sales Cloud page. You can also allow the web application to operate on Sales Cloud data using a JWT token. An external web application can also leverage Oracle Sales Cloud’s Single Sign-On (SSO) authentication. Upon registration, you can reuse a mashup in multiple application pages. You can embed the mashup into the Summary tab or as a new subtab.

By creating mashups, you can do the following:

• Embed and view third-party web applications directly in an Oracle Sales Cloud application.
• Enable SSO authentication for third-party applications.

Mashup Types

Application Composer provides two types of mashups for defining a web application URL:

• **Parameter-based**: A basic mashup that lets you define the complete URL of the web application and the URL parameters during registration.

• **Groovy expression**: An advanced mashup that lets you define the domain or subdomain during registration and later define the path and any URL parameters using a Groovy editor while embedding the mashup.

Creating Mashups: Procedure

To integrate a web application with an Oracle Sales Cloud application, you must first create a mashup by registering the web application in Application Composer.
For example: you might register a web application for a web service that your application calls to retrieve tweets from Twitter, in this case, about Yosemite. The URL for this web service’s location might be:

https://api.twitter.com/1.1/search/tweets.json?q=yosemite

You can choose how you want to define the URL for the web application using the following mashup types:

- A **Parameter-based** mashup, which lets you define the complete URL of the web application and the URL parameters.
- A **Groovy expression** mashup, which lets you provide the base URL here with just the host and the port, so that the mashup can be reusable. Later, while embedding the mashup into an actual page, you can write an expression using a Groovy editor to calculate the rest of the URL path. See Embedding Mashups.

⚠️ **Caution:**
- Ensure that the web application you are registering does not have X-Frame-Options: SAMEORIGIN set in the response header. This option prevents the browser from displaying web applications that are not hosted in the same domain as the parent page.
- If the protocol of your application page is HTTP, then use HTTP while registering the web application. If the protocol of your application page is HTTPS, then use HTTPS while registering the web application.

### Registering a Parameter-Based Mashup

To register a web application using the parameter-based mashup type:

1. Sign in to the Oracle Sales Cloud application as a Sales Administrator.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting **Application Composer** under the Configuration category in the Navigator menu.
4. Under the Common Setup menu, or on the Overview page, click **Mashup Content**.
5. On the Web Applications page, click **Register Web Application**.

The Register Web Application page appears.

6. In the **Name** field, enter a name for the mashup.
   
   This name appears in the Mashup catalog while embedding.
7. Select **Parameter-based**, if not already selected.
8. In the **URL Definition** field, enter the URL of the web application that you want to register.
For example: https://bugs.company.com/path?bug_no=12345678.

9. Optionally, select JWT Access Token to enable the embedded application to call CRM APIs using JWT Tokens. The application uses the token to call REST APIs at the time of launching the web application.

10. In the URL Parameters section, click Add to specify the parameters for the web application.

   📘 Note: The application displays the URL parameters that you add here at the time of embedding the mashup into an actual page, where you can specify the page values for each URL parameter.

11. To delete a parameter, click the delete icon against it.

12. Click Save and Close.

Registering a Groovy Expression Mashup

To register a web application using the Groovy Expression mashup type:

1. Sign in to the Oracle Sales Cloud application as a Sales Administrator.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Under the Common Setup menu, or on the Overview page, click Mashup Content.
5. On the Web Applications page, click Register Web Application.

The Register Web Application page appears.

6. In the Name field, enter a name for the mashup.

   This name appears in the Mashup catalog while embedding.

7. Select Groovy expression.

8. In the URL Definition field, enter the base URL of the web application that you want to register.

   For example: https://en.wikipedia.org/wiki/CRM.
9. Optionally, select **JWT Access Token** to enable the embedded application to call CRM APIs using JWT Tokens. The application uses the token to call REST APIs at the time of launching the web application.

10. Click **Save and Close**.

The Web Applications page displays the newly created mashup. Here, you can edit the mashup by changing its URL definition. You can also choose to set the registered mashup as active or inactive. Note that, while embedding a mashup, the mashup catalog only displays active mashups.

### Editing Mashups: Procedure

You can edit a mashup to change the URL definition of the registered web application and to activate or deactivate the mashup. Note that any changes you make to a mashup will reflect across all the application pages that currently use the mashup.

To edit a mashup:

1. Open the Application Composer by selecting **Application Composer** under the Configuration category in the Navigator menu.
2. Under the Common Setup menu, or on the Overview page, click **Mashup Content**.
3. On the Web Applications page, click the name of the mashup that you want to edit.
4. To change the URL, edit the value in the **URL Definition** field.

5. To activate the mashup, select the **Active** check box; to deactivate it, clear the check box.

   ![URL Definition field](image)

   **Note:** To embed a mashup in an application page, you must ensure that the mashup is active.

6. Click **Save and Close**.

---

### Embedding Mashups: Procedure

You can embed a mashup into an application page either as a new subtab or embed it into the Summary tab. To embed a mashup in an application page, you must ensure that the mashup is active. See **Editing Mashups**.

#### Embedding a Mashup as a New Subtab

To learn how to embed a mashup as a new subtab into an application page, see the **Mashup Content Subtabs** section in the **Subtab Types: Explained** topic.

#### Embedding a Mashup into the Summary Tab

To embed a mashup into the Summary tab of an application page:

1. Sign in to the Oracle Sales Cloud application as a Sales Administrator.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting **Application Composer** under the Configuration category in the Navigator menu.
4. In the Objects navigation tree, expand the object whose pages you want to modify. For example, select the Opportunity object.
5. Click the **Pages** node.
6. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit, or edit another existing layout.
7. Click the Add drop-down button, and then select Mashup Content.

The Select Mashup Content page appears.

8. Select the mashup that you want to embed into your application.

9. Click Insert.

The Embed Mashup Content dialog appears with the configured URL definition.

10. To enable the embedded mashup to expand and collapse at the time of launching the web application, select Enable expand and collapse.

11. Based on the selected mashup type, do one of the following:

   - If the selected mashup is a parameter-based mashup, use the Add Parameters section to add values for each of the web application’s URL parameters configured while registering it. These parameters are appended to the web application’s URL as key-value pairs.
If the selected mashup is a groovy expression mashup, use the Edit Script section to edit the base URL and add to the URL definition of the registered web application.

12. Click **Next**.

The Additional Layouts page appears.

13. Select any other layout that you want to embed the mashup into by moving it from the **Available Layouts** list to the **Selected Layouts** list.

14. Click **Save and Close**.
The embedded mashup appears at the bottom of the Summary tab.

Viewing the Mashup Content: Procedure

To view the content of an embedded mashup:

1. From the home page or the navigator menu, select the application page that contains the embedded mashup. For example, Opportunities.
2. On the landing page, click the name of an object.
   
   For example, if you have selected Opportunities, then click the name of an opportunity from the list.

   Based on where the mashup is embedded, the application displays the mashup content in the Summary tab or as a new subtab.
The following examples illustrate both the scenarios:

- A mashup embedded into the Summary tab of an application page:

```
Edit Opportunity: test : Summary
```

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Quantity</th>
<th>Estimated Price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Wiki

```
CRM
From Wikipedia, the free encyclopedia

CRM may refer to:

Sales and marketing
```

Not logged in Talk Cont

```
Main page
Contents
Featured content
```
• A mashup embedded into an application page as a new subtab:

**Modifying Work Area Lists**

**Creating Work Area Lists for Your Organization**

**Watch:** This tutorial shows how sales administrators can create their own work area lists (saved searches) for specific job roles or for the sales organization as a whole. The content of this video is also covered in text topics.

**Hiding and Deleting Work Area Lists**

**Watch:** This tutorial shows how you can delete administrator-defined lists and hide lists provided by Oracle in the different Oracle Sales Cloud work areas. The content of this video is also covered in text topics.
Procedures

Sales administrators can use the procedures in this topic to create their own work area lists, also called saved searches. You can create lists to replace those supplied by Oracle and make your lists available to all users or target users with specific job roles.

**Note:** If you are creating lists for a specific job role, then you must first provision yourself with that job role so you can test your work before publishing the sandbox. See the steps outlined in the related topic Assigning Yourself an Additional Job Role.

To modify the work area lists, you do the following:

1. Navigate to the work area and get ready by activating a sandbox and opening Page Composer.
2. Make your modifications. You can:
   - Create your own list.
   - Hide lists provided by Oracle.
   - Delete any of the lists you created.
3. Publish the sandbox to make your changes available to users.

Getting Ready to Modify Work Area Lists

Follow these steps to enable a sandbox and get ready to start modifying lists:

1. Click your user image or name, select **Manage Sandboxes**, and activate an existing sandbox or create and activate a new one.
   
   The application displays a bar across the top of the window indicating the sandbox is active.
2. Navigate to the work area landing page you want to modify.
3. From the same user name menu, select **Customize Pages**.

   The Customize Pages window appears.
4. Select one of the two context layers:
   - **Site** to make your changes available to everyone.
   - **Job Role** to target a specific job role you select from the list.

   The list of available job roles includes all the job roles assigned to you directly and any job roles that are inherited by those job roles.

   The Editing: User Interface Page Composer toolbar appears at the top of the page with the title Editing: User Interface and the **Add Content** mode selected.
The following figure shows a screen capture of the sandbox bar (callout 1) at the top of the page and the Page Composer toolbar underneath (callout 2).

You are now ready to modify the work area lists. After you make your changes and test them, you must publish the sandbox to make your changes available to your users.

Creating a List

With both the sandbox and the Editing: User Interface Page Composer toolbar displayed on the top of the page, follow these steps to create your list. You create a new list by editing an existing list and saving it under a new name.

1. In the work area landing pad, click Show Advanced Search next to the List field.
   
   The Advanced Search panel appears.

2. From the Saved Search list, select a saved search to use as the starting point for creating a new one.

   Tip: To create a list using only one field, including administrator-created fields, select a search with either Name or Close Date in the title. For opportunities and leads, select the Close Date saved search. For all other objects, select the name saved search, for example, the Account Name saved search or the Contact Name saved search.

3. Make your changes. You can:

   - Select a different record set to change the scope of your search. For example, selecting My territory hierarchy searches all the records in your territories and their subordinate territories. The available record sets vary from object to object.
     
     To improve saved search performance, restrict your saved searches to smaller record sets. For example, rather than searching all the records you can see, search all the records in your territory hierarchy. Or restrict your searches to a smaller geographical area. For example, search all the accounts in one state instead of the whole country.
   - Add additional fields by clicking Add.

   Note: If you are adding an administrator-created field to your search, then the field must be indexed for best search performance.
Select different operators for the fields in your search. While most of the operators, which differ field by field, are self-explanatory, here are some notes on the more complex:

- **Between**

  Selecting the Between operator for a date field, prompts you to enter a specific date range for the search.

- **Rolling-time operators**, including *Yesterday, Today, Tomorrow, Last Week, This Week, Next Week, Last Month, This Month*, and *Next Month*

  Searching for opportunities created this month, for example, returns opportunities created in the current calendar month. The rolling-time filters use the time zone of the signed-in user for the calculation.

- **Current User**

  In some fields, including the Last Updated By and Created By fields, you can create a list that displays only the records relevant to each signed-in user. For example, when you create a list of all opportunities last updated by the current user, all of your users can view the opportunities they recently updated.

- **Is Blank, Is Not Blank, and Does Not Equal**

  Selecting the Is Blank operator, makes it possible to search for records missing values in a text field or a field validated by a list of values, for example. Administrators can make these operators available on fields that are not mandatory, by setting two system profile options. The Enable Additional Search Operators for Text Fields (ZCA_ENABLE_ADDITIONAL_TEXT_OPERATORS) profile option enables the Is Blank and Is Not Blank operators in text field searches. The Enable Additional Search Operators for List of Values (ZCA_ENABLE_ADDITIONAL_LOV_OPERATORS) enables the Is Blank, Is Not Blank, and Does Not Equal operators on fields validated by list of values (both fixed choice and dynamic choice list fields).

  - Specify which attributes you want to display in the search results table by selecting **Columns** from the **View** menu.
    
    You can select specific columns or display all columns.

  - Reorder the filter conditions by clicking **Reorder**.

  - Delete any fields you added to the search.

    You cannot delete the fields provided by Oracle. You can only delete fields you added.

4. You can test your search by clicking **Search**.

5. When you are satisfied with the results, click **Save**.

   The Create Saved Search window appears.

6. Enter a new name for the saved search.

   You cannot reuse the names of the saved searches provided by Oracle with the application.

7. Make sure the **Run Automatically** option remains selected. Selecting this option runs the query each time you select the list in the UI.

8. If you want users to see the list generated by this search when they navigate to the work area, then select the **Set as Default** option.

   **Note:** Making a search the default does not override any default searches individual users may have created for their own use. Each user can personalize lists provided by administrators.

9. Click **OK** to return to the work area landing page.
Hiding a Saved Search Provided by Oracle

Use the following procedure to hide a saved search supplied by Oracle from the list of searches available for selection in the work area. If you want to hide a saved search that is designated as the default search, then you must first designate a different list as the default.

1. In the work area landing pad, click **Show Advanced Search** next to the **List** field.

   The Advanced Search panel appears.

2. From the **Saved Search** list, select a saved search different from the one you want to hide.

3. Now select **Personalize**.

   The Personalize Saved Searches window appears.

4. Select the saved search you want to hide.

5. Deselect the **Show in Search List** option.

6. Click **OK**.

   A saved search supplied by Oracle remains available for future use but does not display in the work area. You can restore the saved search to the list in the future by selecting the **Show in Search List** option.

Deleting a Saved Search You Have Created

Use the following procedure to delete a saved search you have created.

> **Note:** If you delete a saved search that you specified as the default, then no list displays when users navigate to the work area.

1. In the work area landing pad, click **Show Advanced Search** next to the **List** field.

   The Advanced Search panel appears.

2. From the **Saved Search** list, select a saved search you want to delete.

3. Now select **Personalize**.

   The Personalize Saved Searches window appears. The **Saved Searches** list displays the name of the saved search you are about to delete.

4. Click **Delete**.

5. Click **OK**.

Related Topics

- Setting Up Sandboxes: Procedure
- What gets saved when I create a saved search for searches with multiple criteria?
- Assigning Yourself Additional Job Roles Required for Testing

FAQ for Extending Simplified Pages
What's the difference in upgrade behavior between standard layouts and duplicated layouts?

After an upgrade, the standard layouts for an object’s creation and details pages automatically include any new underlying changes made as part of the upgrade.

However, duplicated layouts (copied versions of standard layouts) exist independently from the original, standard layout, and in general, do not uptake most underlying changes that come with an upgrade. In other words, duplicated layouts remain functionally identical to their preupgrade versions, even after an upgrade.

Thus, after every upgrade, carefully review all your duplicated layouts. You can re-create them if you want to pick up new changes introduced in the upgrade.
6 Extending Mobile Pages

Overview

Read this chapter to learn how to extend Oracle CX Cloud Mobile and Oracle Sales Cloud Mobile.

In this chapter, you will learn:

• How to use Application Composer to make objects and fields visible in the mobile application without having to perform specific application changes for any particular device
• How to add an Oracle Business Intelligence report to your mobile application
• How to test mobile application changes using a sandbox

Oracle CX Cloud Mobile

Configuring Oracle CX Cloud Mobile: Explained

You can configure the Oracle CX Cloud Mobile iPhone and Android applications for your organization’s particular requirements using Application Composer. Using Application Composer, you can manage which objects and fields are visible on the Oracle CX Cloud Mobile application, without having to carry out specific configurations for a particular device.

Before starting your configurations, you will need to set the ZMS_DISABLE_OSCM profile option to ENABLED. Setting the profile option will enable the new mobile interface designer in Application Composer. For more information about profile options, see the profile options chapter in the Implementing Sales guide.

Creating a Page Layout for a Feature

You can create a List, Detail, or Edit page layout for a Oracle CX Cloud Mobile feature. Creating your own layout enables you to select the fields your users will see for a feature’s views. See the Creating a Page Layout for a Feature: Procedure topic for details about how to create a page layout.

When you create a page layout, you can add, remove, move, and change a field’s display format using the mobile interface designer. Find out more information about using the mobile interface designer in the following FAQs:

• How can I add a field to a feature’s page layout?
• How can I delete a field in a feature’s page layout?
• How can I move a field in a feature’s page layout?
• How can I edit a field’s display format in a feature’s page layout?
• How can I hide or display user actions for a feature?

Adding a Role to a Page Layout

You can add a role to an application feature’s List, Detail or Edit layout. For example, a user with the Sales Manager role might want to see certain fields on an opportunity detail record that other sales team members won’t need. See the Adding a Role to a Layout: Worked Example topic for details about adding a role.
Creating Criteria for a Page Layout
You can create criteria to define a set of conditions that have to be met before the page layout is displayed for a feature’s Detail or Edit views. See the Creating Criteria for a Feature Page Layout: Procedure for step by step instructions.

Adding Your Own Object to a Page Layout
You can add your own objects to your CX Cloud Mobile application, and add page layouts in the same way that you can with standard sales objects (or features, as they are known as in the Mobile Application Setup). See the Adding Your Own Object to a Page Layout: Procedure topic for more details.

Assigning Geographical Regions to a Page Layout
You can assign geographical regions to a page layout, which will restrict a page layout’s availability to users from your selected set of geographical regions.

Other CX Cloud Mobile Configurations
You can configure the application in many other ways, such as creating saved searches and adding Oracle Business Intelligence reports to a feature’s Analytics tab. See the Oracle Sales Cloud Implementing Sales guide, or the Oracle Sales Cloud Extending Sales guide for more details.

Testing Configurations
After you have configured Oracle CX Cloud Mobile using Application Composer, you should test your configurations before distributing them to your user’s mobile devices. See the Testing Oracle CX Cloud Mobile Configurations: Worked Example topic for more details.

Creating a Page Layout for a Feature: Procedure
This procedure shows you how to create a List, Detail, or Edit page layout for an Oracle CX Cloud Mobile feature. Creating your own layout enables you to select the fields your users will see for a feature’s views in CX Cloud Mobile.

Tip: You can also specify which user roles can view a particular layout, create criteria that have to be met to display the layout, add custom objects, and specify which geographical regions can view the layout.

1. Sign in to the Oracle Sales Cloud application as user with a Sales Administrator or Sales Implementor role.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Under the Common Setup Menu, or on the Overview page, click Mobile Application Setup.
5. In the Application Feature pane, click the feature that you want to create a layout for.
6. Select the relevant type of page view, such as the detail or list view.
7. In the Layout pane, click the Duplicate icon for the Standard layout and enter a layout name.
8. Click OK.
9. Select the Active tick icon if it’s not already ticked.
10. Add fields to the layout by selecting the field in the Available Fields pane, and then move the field onto the mobile interface designer. See the topic called How can I edit a field’s display format in a feature’s page layout for information about defining a display format for your fields.
11. You can remove fields by clicking the field you want to delete on the mobile interface designer and then clicking the cross icon.
12. You can move fields by clicking the field and moving it to your preferred location.
13. Click **Save** in the top right hand side of the Application Composer page.

14. Test and publish your new page layouts. See the Testing Oracle CX Cloud Mobile Configurations: Worked Example topic for details about how to check your mobile configurations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud Extending Sales guide.

### Related Topics
- Assigning Geographical Regions to a Page Layout: Procedure

---

**How can I add a field to a feature's page layout?**

Navigate to the mobile interface designer in Application Composer, select the field in the Available Fields pane, and then move the field onto the mobile interface designer. Click **Save** in the top right hand side of the Application Composer page. See the Creating a Page Layout for a Feature: Procedure topic for more information about creating a page layout.

**How can I delete a field in a feature's page layout?**

Navigate to the mobile interface designer in Application Composer, select the field you want to delete, and then click the cross icon. Click **Save** in the top right hand side of the Application Composer page. See the Creating a Page Layout for a Feature: Procedure topic for more information about creating a page layout.

**How can I move a field in a feature's page layout?**

Navigate to the mobile interface designer in Application Composer, select the field you want to move, and then click the field and move it to your preferred location. Click **Save** in the top right hand side of the Application Composer page. See the Creating a Page Layout for a Feature: Procedure topic for more information about creating a page layout.

**How can I edit a field's display format in a feature's page layout?**

Navigate to the mobile interface designer in Application Composer, select the field you want to edit, and click the **Edit** pencil icon. Select the display format you want and click **Save** in the top right hand side of the Application Composer page. See the Creating a Page Layout for a Feature: Procedure topic for more information about creating a page layout.

> **Note:** Not all fields can be edited. You can only edit fields that require you to choose a display format.

---

**Adding a Role to a Layout: Worked Example**

This example describes how to add a role to an Oracle CX Cloud Mobile page layout. You can add a role to an application feature's List, Detail or Edit layout. For example, a user with the Sales Manager role might want to see certain fields on an opportunity detail record that other sales team members won't need.

> **Note:** You can’t add a role to a custom feature. Custom features are created when you create a custom object in Application Composer, and they’re selected from the Available Features pane.
In this example, you add a role to an Opportunity Detail layout and display the new layout in the CX Cloud Mobile application.

### Add a Role to a Layout

1. Sign in to the Oracle Sales Cloud application as user with a Sales Administrator or Sales Implementor role.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Under the Common Setup Menu, or on the Overview page, click Mobile Application Setup.
5. In the Application Feature pane, click the feature that contains the page layout that you want to add roles to. In this example, expand the Opportunities subheader.
6. Select the relevant type of page view, such as the detail or list view. In this example, select the Detail view.
7. In the Layouts pane, select the relevant page layout, or create a new page layout (see the Creating a Page Layout for a Feature: Procedure topic for details about how to create a page layout).
8. In the Assigned Roles pane, select the roles you want to add and click Save.

### Test and Publish Your Changes

1. Check that your changes appear in the mobile application. See the Testing Oracle CX Cloud Mobile Configurations: Worked Example topic for details about how to check your mobile configurations.
2. When you are happy with your changes, publish your sandbox to distribute your configurations to all CX Cloud Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Extending Sales guide.

### Related Topics
- Using Sandboxes: Explained

### Creating Criteria for a Feature's Page Layout: Procedure

This procedure shows you how to create criteria for an Oracle CX Cloud Mobile feature's page layout. Creating criteria enables you to define a set of conditions that have to be met before the page layout is displayed for a feature’s Detail or Edit views. For example, if you create criteria for an Opportunity Detail layout as follows: Win probability is Greater than 50%, then any opportunities with a win probability greater than 50% will use your layout in the Detail view.

> **Note:** You can’t create criteria for a feature's List view.

1. Sign in to the Oracle Sales Cloud application as user with a Sales Administrator or Sales Implementor role.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Under the Common Setup Menu, or on the Overview page, click Mobile Application Setup.
5. In the Application Feature pane, click the feature that contains the page layout that you want to add criteria to.
6. Select the relevant type of page view, such as the detail or list view.
7. In the Layouts pane, select the relevant page layout, or create a new page layout (see the Creating a Page Layout for a Feature: Procedure topic for details about how to create a page layout).
8. In the Advance Criteria pane, click Add.
9. Create your criterion by selecting a field, operator, and then entering the relevant field value.

> **Note:** You can’t select a field value from a list of values, so you will need to type in the value if you would normally select it from a list of values.
10. To add a conditional statement click Add and select AND or OR. Enter the field, operator, and relevant field value.

11. When you have finished creating your criteria for the page layout, click Save in the top right-hand side of the Application Composer page.

12. Test and publish your new page layout criteria. See the Testing Oracle CX Cloud Mobile Configurations: Worked Example topic for details about how to check your mobile configurations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud Extending Sales guide.

Adding a Custom Object to a Page Layout: Procedure

This procedure shows you how to add a custom object to your Oracle CX Cloud Mobile application. Sales objects are also known as features in CX Cloud Mobile, and when you create a custom Sales object in Application Composer a corresponding feature is also created in CX Cloud Mobile. These features can be found in the Available Features pane in the Mobile Application Setup page.

Here are the steps to add a custom object, or feature, to CX Cloud Mobile:

1. Sign in to the Oracle Sales Cloud application as user with a Sales Administrator or Sales Implementor role.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Select the Sales application and find the custom object you want to add to CX Cloud Mobile.
5. Expand the object’s menu and click Fields.
6. Select the field you want to add to CX Cloud Mobile.
7. In the Constraints section, enable the Include in Service Payload option.
8. Click Save and Close.
9. Repeat steps 6 to 8 for each field that you want to add to the application.
10. Next, under the Common Setup Menu, click Mobile Application Setup.
11. In the Available Feature pane, click on the custom object (otherwise known as a feature) that you want to add and move it onto the mobile interface designer.
12. Click OK in the information message about the views that will be created.
13. If you want to add custom layouts to the List, Edit, or Detail views, see the Creating a Page Layout for a Feature: Procedure topic for more details.
14. If you want to add fields to the picker, click on the fields you want to add in the Available Fields pane, and move them onto the mobile interface designer.
15. When you have finished adding features, click Save in the top right-hand side of the Application Composer page.
16. Test and publish your new features. See the Testing Oracle CX Cloud Mobile Configurations: Worked Example topic for details about how to check your mobile configurations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud Extending Sales guide.

Configuring the Call Report Pages: Explained

You can configure the Call Report pages within Oracle CX Cloud Mobile using Application Composer. Using the mobile interface designer within Application Composer, you can manage the fields on the Call Report detail and edit pages. You can add, remove, and move fields, and edit a field’s display format.

Here’s how to navigate to the Call Report feature, so that you can configure it using the mobile interface designer:

1. Sign in to the Oracle Sales Cloud application as a user with a Sales Administrator role.
2. Select the sandbox you want use for your configurations.
3. Open Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
4. Select the Sales application.
5. Under the Common Setup Menu, or on the Overview page, click Mobile Application Setup.
6. In the Application Feature pane, click the Call Report application feature.
7. Configure the feature using the steps outlined in the Configuring Oracle CX Enterprise Mobile: Explained topic and in the page layout FAQs.

Adding Oracle Business Intelligence Reports to a Feature's Analytics Tab: Procedure

To view Oracle Business Intelligence Analysis (BI) reports in a feature’s Analytics tab (for example, an Accounts or Contacts Analytics tab), you must first perform some configuration tasks.

1. Sign in to Oracle Sales Cloud as a user that has a Sales Cloud Administrator job role and create, or activate, a sandbox. See the topic about using sandboxes for more information about creating, activating, and publishing sandboxes.
2. Navigate to Application Composer.
3. Select Mobile Application Setup under the Common Setup list.
4. In the Application Features pane, click the feature that you want to add the report to.

Note: You can add reports to the following Sales objects only: Account, Opportunity, Contact, Lead, Partner, and Deal Registration.
5. Click Analytics.
6. In the Available Reports section, search for the report you want to add to the Sales object’s Analytics tab.

Note: You can view all of your BI reports that you have available in your catalog.
7. Move the report onto the mobile interface designer.
8. Click on the report on the mobile interface designer to see the Report Information, Report Filters, and the Assigned roles sections.
9. In the Report Filters section you can see the filters that have been created for the BI report. To further filter the report, or to enable contextual reporting, click the report in the mobile interface designer, and enter the parameters you want the mobile report to use in the Report Filters section. The parameter values are the attribute names which are available in Application Composer under the standard and custom fields list. You can override an BI report parameter value so that only the data relevant to the Sales object you are viewing is displayed. Any attributes (apart from ID and Name) can be used as filters by passing the appropriate parameters. For example, for an Account report you could filter using the @PartyId or @PartyUniqueName parameters to see report information relevant to the Account you are viewing.

Note: You can only provide parameter values to filters that have already been defined in the original BI report.
10. Click the filter’s check box to make the filter active.
11. In the Assigned Roles section, select the roles that will be able to view the report. If you don’t assign a role then the report will be available to all job roles.
12. Click Save.
13. Check that your changes appear in the mobile application. See the Testing Oracle CX Cloud Mobile Configurations Using a Sandbox: Worked Example topic for details about how to check your mobile configurations.

14. When you are happy with your changes, publish your sandbox to distribute your configurations to all CX Cloud Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud Extending Sales guide.

Related Topics
- Using Sandboxes: Explained

Testing Oracle CX Cloud Mobile Configurations: Worked Example

After you have configured Oracle CX Cloud Mobile using Application Composer, you should test your configurations before distributing them to your user’s mobile devices.

Task Summary
To test your configurations, complete the following tasks:

1. Download the configurations from the sandbox and verify them.
2. Publish the sandbox

Download Configurations From the Sandbox

1. After making your configurations, keep Oracle Sales Cloud open (signed in with a Sales Administrator or Sales Implementor role). Make sure that the sandbox where you made the changes is active in the application.
2. Open CX Cloud Mobile on your mobile device and sign in as a user that will be able to view your configurations. For example, if the changes have been made to a layout with an assigned role, then you’ll need to sign in with a user that has the required role permissions to see the changes.
3. Tap the menu button (on the top left hand side of the page) and tap Sandbox to download the configurations from the sandbox.
4. Select the sandbox that contains your configurations and tap Save.
5. Check your configurations. When you’re happy with your configurations you need to publish the sandbox.

Publish the Sandbox

1. Sign in to Oracle Sales Cloud as the same user you used to make the configurations.
2. Publish your sandbox to distribute your configurations to all Oracle CX Cloud Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Extending Sales guide.

Oracle Sales Cloud Mobile

Oracle Sales Cloud Mobile Extensibility: Explained

Application Composer lets implementors configure the Oracle Sales Cloud Mobile Sales iPhone, Android and BlackBerry applications. Using Application Composer, implementors can manage which objects and fields are visible on the Oracle Sales Cloud Mobile application without having to carry out specific configurations for a particular device.
Implementors can manage the following for the Oracle Sales Cloud Mobile application:

- Enable standard Oracle Sales Cloud Sales and Common objects that are not enabled by default for smartphones.
- Enable custom Sales and Common objects for smartphones.
- Change the fields (including custom fields) visible on Oracle Sales Cloud Mobile for mobile-enabled Sales and Common objects (standard or custom objects).
- Configure the Sales Cloud Mobile layout based on roles, record type, expression, or any combination thereof.
- Add Business Intelligence reports to the Sales Cloud Mobile application.

Configuring Oracle Sales Cloud Mobile

This procedure shows you how to configure Oracle Sales Cloud Mobile for a specified sales object using Application Composer. For more information about Application Composer, see the Oracle Sales Cloud - Extending Sales guide.

First you must open or create a sandbox, and then you configure Oracle Sales Cloud Mobile using Application Composer.

Opening or Creating a Sandbox

1. Sign in to Oracle Sales Cloud with a user that has a Sales Cloud Administrator job role.
2. In the global region, expand the Settings and Actions menu, then select Manage Sandboxes, under the Administration subheading.
3. Select the sandbox that you want to use to make your configurations. You might need to make a sandbox active, or create a sandbox, if a suitable sandbox does not exist.

Configuring Oracle Sales Cloud Mobile Using Application Composer

1. Open Application Composer by using the Navigator menu, and selecting Application Composer under the Configuration category.
2. Select the custom or standard object you want to configure.
3. Select the Pages node in the navigation tree.
4. Select the Mobile Pages tab to see the mobile configuration options for the parent and its child objects.
5. Click either Create <object name>, or Edit <object name>, depending on whether the object is already enabled for Sales Cloud Mobile. You can also create or edit the child objects that appear under the main object.
6. Navigate through the configuration wizard to configure your selected object.

Configuring Your Mobile Springboard

You can add or remove object icons from the Sales Cloud Mobile application as follows:

Adding or Removing object icons

1. Within Application Composer, choose Sales from the Applications menu.
2. Open the Mobile Application Setup page, either by clicking the page link in the Common Setup menu, or by clicking the page link in the Overview section of the Application Composer page.
3. Click Configure Springboard.
4. Select the object you want to add or remove, and click the arrows pointing toward the right or left, moving the object to the Available list, or Selected list.
5. Click Save and Close.
Dynamic Choice Lists in Oracle Sales Cloud Mobile: Explained

A dynamic choice list field provides a list of values from which your users can select a value at run time. The list of values is considered dynamic because the list is based on a query of a related object's records. Some dynamic choice list fields are standard, which means they are provided automatically with Oracle Sales Cloud. You can also create user-defined dynamic choice list fields in Application Composer. Both standard and user-defined dynamic choice list fields can be displayed on Oracle Sales Cloud pages, as well as on Oracle Sales Cloud Mobile pages.

Read this topic to learn about the following:

- What are dynamic choice list fields, and why are they useful?
- Adding dynamic choice list fields to Oracle Sales Cloud Mobile pages.
- Creating the related object subtabs in Oracle Sales Cloud Mobile.

What's a Dynamic Choice List Field?

A dynamic choice list is a field that contains a list of values which are populated from the actual data of a related object. At run time, your users can select a value from the field. This selection associates that related object's value with the primary object's record that the users are currently looking at.

For example, let's say your users need to specify an account for an agreement:

- **Desired result:**
  On the Create Agreement or Edit Agreement page, you want your users to select an account from the Account Name field.

- **Steps to enable the desired result:**
  To enable this behind the scenes, use Application Composer to create the Account Name dynamic choice list field on the Agreement object. The Account Name field will be populated with a list of account records from the Account object. Once that field exists on the Agreement object, add the field to all Agreement pages (including Mobile pages).

- **End result:**
  The Account Name field will now be available from the Create Agreement or Edit Agreement pages. The field itself will include a list of actual account names, populated from the Account object.

Why are Dynamic Choice List Fields Useful?

Dynamic choice list fields are useful because they display at run time with a prepopulated list of values, which your users can pick from. But, dynamic choice lists fields are also very useful because, behind the scenes, they enable a one-to-many relationship between the source object and target object.

This means that not only do you get the ability to associate a source object record (using our previous example, an account) to a target object record (an agreement) using the dynamic choice list field. But, you can also add a related object subtab to the source object’s details page (the Account details page), showing a list of all the target object records (agreements) that are associated with a single source object record (account).

In our previous example of making a list of accounts available for association with an agreement, the relationship that is created between the Account and Agreement objects is a one-to-many relationship, where one account can be associated with multiple agreements. Behind the scenes, an account identifier is stored in the Agreement object's table.

Once the dynamic choice list field is created, the one-to-many relationship that is automatically created means that you can now use Application Composer to display an Agreements subtab on the Account details page. This subtab lists all the
agreements that are related to an account. Having this ability to add subtabs is a nice benefit; in a single view, your end users can see all the agreements that are related to a single account. Creating subtabs based on an existing dynamic choice list field is discussed below.

Adding Dynamic Choice List Fields to Oracle Sales Cloud Mobile Pages

You can add standard and user-defined dynamic choice list fields to your Oracle Sales Cloud Mobile pages. Adding dynamic choice list fields to Mobile pages requires three steps:

1. Configure the Mobile picker page for the source object that populates the dynamic choice list field. This is a required, one-time configuration task per object. If you don't configure the picker page, then you won't be able to display that object's dynamic choice list fields on any Mobile UI. This configuration is required for both standard objects as well as custom objects.

   Note: Some exceptions exist.
   - For example, the Account object is delivered with the picker page already configured for your use, so you don't have to configure a picker for the Account object.
   - Some other standard objects don't support the Mobile picker page. In this case, if standard dynamic choice list fields exist based on those objects, then you won't be able to add those fields to Mobile UIs.

To configure the picker page for standard and custom objects (except Account):

   a. Navigate to Application Composer.
   b. Under the Objects navigation tree, expand the tree structure for your object.
   c. Click the Pages node.
   d. Click the Mobile Pages tab.
   e. In the Picker region, click the Create Mobile Picker link for your object.
   f. On the Configure Mobile Picker page, select the fields that you want to display in the picker page. For example, let’s say this is the picker page where your users will search for a contact. In addition to the contact name, you might also want to display the contact city in the picker page. At run time, your users will be able to decide between Mary Smith from New York, or Mary Smith from Los Angeles.
   g. Click Save and Close.

2. After the Mobile picker page is created for an object, you can now create any dynamic choice list field based on that same object. See: "Dynamic Choice Lists: Explained".

3. Once your dynamic choice list field is created, you can now add the field to your Mobile pages. See: "Configuring Oracle Sales Cloud Mobile".

   Tip: If your dynamic choice list field is not available to add to a Mobile UI, then confirm that the Mobile picker page was created for the dynamic choice list field’s source object. See Step 1 above.

   Note: Navigating to a sales object’s details page after selecting the sales object from a DCL field is not supported currently.
Adding Subtabs to Oracle Sales Cloud Mobile Pages

After you create a one-to-many relationship between objects using a dynamic choice list field, you can then expose the "many" object's records on a subtab that is displayed on the "one" object's details page. You do this by creating a related object subtab in Application Composer.

Note: You can display a custom object subtab on a custom object Mobile details page. You can also display a custom object subtab on a standard object Mobile details page. Displaying a standard object subtab on a custom object Mobile details page, however, is not supported.

Adding a subtab to Mobile pages requires four steps:

1. Create and configure the Mobile relationships list for the related object.
2. Configure the picker page for the related object.
3. Indicate if you want your end users to be able to create new records and add existing records, directly from the subtab.
4. Finally, add the subtab to the Mobile details page layout or layouts where you want the subtab to appear.

Let's review each step in depth:

1. Create and configure the Mobile relationships list for the related object. This is where you create the table format that displays on the subtab.
   a. Navigate to Application Composer.
   b. Under the Objects navigation tree, expand the tree structure for your object.
   c. Click the Pages node.
   d. Click the Mobile Pages tab.
   e. In the Related Objects region, view the list of objects that are available to add as subtabs to your object's details page. Click the Create Mobile Page link for your related object. The Create Mobile Page link is enabled if the related object's top level Mobile pages are already created.
   f. On the Create or Edit List Layout page, use the List View region to indicate which related object fields you want to appear on the subtab.

2. Configure the picker page for the related object. If you enable your end users to add one or more existing records to the subtab at run time, then they will access this picker page.
   a. On the Create or Edit List Layout page, use the Picker: Select many region to indicate which related object fields you want to appear on the picker page for the related object.

3. Indicate if you want your end users to be able to create new records and add existing records, directly from the subtab.
   a. On the Create or Edit List Layout page, check Show Add and Show Create to enable those actions on the subtab.
   b. Click Save and Close.

4. Finally, add the subtab to the Mobile details page layout or layouts where you want the subtab to appear.
   a. Back on the Mobile Pages tab, navigate to the Detail Page Layouts region.
   b. Duplicate the standard layout to create a new layout to edit, or edit another existing layout.
   c. In the Related Objects region, view the list of related objects that are available to add as subtabs to your object's details page. In the Available Related Objects list, your subtab displays using the one-to-many relationship name that was automatically created when you first created the dynamic choice list field. To display your subtab on the Mobile details page, move that subtab to the Selected Related Objects list.
d. Click Save and Close.

Related Topics

- Dynamic Choice Lists: Explained
- Object Relationships: Explained
- Configuring a Search and Select Dialog Box: Explained

Displaying a List of Products for Revenue Items

You can enable users to add products to a revenue item, by following these setup steps:

1. Within Setup and Maintenance, search for the Manage Product Group Usage task.
2. Click the Go to button for the Manage Product Group Usage task.
3. Under Product Group Usage, select the Base row.
4. Select the Miscellaneous tab and click on the Add icon.
5. Create a New Row with the following values:
   - Name: Hide Products
   - Value: No
   - Mode: Oracle Fusion Mobile Sales

This displays a list of Products which enables users to attach a Product to a Revenue Item. If required, users will still be able to attach a Product Group to a Revenue Item, as well as a Product. If a Product list is not displayed then a user will only be able to attach a Product Group to a Revenue Item.

How can I set up the automatic password saving on smartphones and tablets?

Search for the Manage Administrator Profile Values task in the Setup and Maintenance work area. Set the Password Save on Phone Enabled profile option to either Y (Yes) or N (no).

Setting Up RSS Feeds

You can set up RSS feed sources so that the RSS feeds will appear at the bottom of the Sales Cloud Mobile pages, together with related objects, as follows:

1. Within Setup and Maintenance, search for the Manage Administrator Profile Values task.
2. Find the Manage Administrator Profile Values task and click Go to Task.
3. In the Profile Display Name field, enter RSS Feed Source and click Search.
4. Add RSS feed URLs as required.
Testing Oracle Sales Cloud Mobile Configurations Using a Sandbox: Worked Example

Test all of your configurations in a sandbox before publishing them to the main application. Sandboxes are standalone environments where you define and test configurations before deploying the configurations to the production application. Use Application Composer to configure Oracle Sales Cloud Mobile pages or objects in a sandbox environment, and then view your configurations on your mobile device prior to publishing the changes.

Opening a Sandbox

1. Sign in to Oracle Sales Cloud as a user who has a Sales Cloud Administrator job role.
2. In the global header, expand the Settings and Actions menu.
3. Select Manage Sandboxes, under the Administration subheading.
4. Select the sandbox in which you want to make your configurations.

Note: You might have to make a sandbox active or create a new sandbox.

Configuring Oracle Sales Cloud Mobile Using the Application Composer

1. Open the Application Composer by selecting Application Composer under the Configuration category in the Navigator menu.
2. Select the parent object that you want to configure.
3. Select the Pages node in the navigation tree.
4. Select the Mobile Pages tab to see the mobile configuration options for the parent and its child objects.
5. Configure the mobile pages as you want.

Note: If you want to configure a layout for a specific role, record type, or expression, you must first duplicate an existing page, configure that page, then specify a role, record type or expression for that page layout.


Checking Your Configurations in the Oracle Sales Cloud Mobile Application

1. Sign in to Oracle Sales Cloud application as an Oracle Sales Cloud Mobile user who has a Sales Representative, Sales Manager, or Sales Vice President job role.
2. Select the sandbox that contains your configurations.
3. Keeping the Oracle Sales Cloud browser window open on your laptop or PC, open Oracle Sales Cloud Mobile on your mobile device and sign in using the same user you used to sign in to Oracle Sales Cloud.

Note: Signing in to the Oracle Sales Cloud Mobile application as the same user in step 1 lets you view the sandbox you selected in step 2. If the Oracle Sales Cloud browser window is still open, only you can access the sandbox on the Oracle Sales Cloud Mobile application. All other users can only view the published version of the application.

4. Check the pages that you configured to ensure that they’re working as expected.
5. Publish your sandbox to distribute your configurations to all Oracle Sales Cloud Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Extending Sales guide.
Related Topics

- Sandboxes: Explained
- Using Sandboxes: Explained
- Sandboxes: How They Work with Application Changes and Features
7 Using Groovy Scripts

Overview

Groovy is a standard, dynamic scripting language for the Java platform. You write Groovy scripts using Application Composer’s expression builder, which appears in many places as you modify existing objects or create new custom ones. Read this chapter to learn about how and where you can use Groovy scripting in Application Composer.

To fully understand all the scripting features available to you in Application Composer, you should also review the Oracle Sales Cloud Groovy Scripting Reference.

In this chapter, you will learn about:

- Where you can use Groovy in your application, along with examples of one or more lines of Groovy code
- How to access view objects using the newView() function, for programmatic access to object data
- How to create global functions, which is code that multiple objects can share
- How to call Web services from your Groovy scripts. You might call a Web service for access to internal or external data, or, for example, to perform a calculation on your data.

What kind of scripts will you write?

You write Groovy scripts using Application Composer’s expression builder, which appears in many places throughout Application Composer as you modify existing objects or create new custom ones.

- You will write shorter scripts to provide an expression to calculate a custom formula field’s value or to calculate a custom field’s default value, for example.
- You may write somewhat longer scripts to define a field-level validation rule or an object-level validation rule, for example.

Additional examples of where you write Groovy scripts in Application Composer are described in "Groovy Scripting: Explained."

To learn more about how to best use the features available in the expression builder when writing scripts, see "Groovy Tips and Techniques" in Oracle Sales Cloud Groovy Scripting Reference.

Related Topics

- Groovy Scripting: Examples

Groovy Scripting: Explained

Groovy is a standard, dynamic scripting language for the Java platform for which Application composer provides deep support. This topic provides an overview of where you can use groovy in your application and gives some samples of one or more lines of Groovy code. You can also find information on Supported Classes and Methods for Use in Groovy Scripts and some examples in the Related Topics section.
For more information on groovy scripting, see Oracle Sales Cloud Groovy Scripting Reference at http://www.oracle.com/pls/topic/lookup?ctx=cloud132&id=CGSAC.

> **Note:** Read "Supported Classes and Methods for Use in Groovy Scripts", which documents the only classes and methods you may use in your Groovy scripts. Using any other class or method will raise a security violation error when you migrate your code to later Oracle Sales Cloud maintenance releases. Therefore, it is strongly suggested that the Groovy code you write uses only the classes and methods shown there to avoid the time-consuming task of having to rewrite your code in the future.

### Groovy Scripting Terminology Explained

Throughout the document the term script is used to describe one or more lines of Groovy code that the Oracle ADF framework executes at run time. Often a very-short script is all that is required.

For example, to validate that a Commission Percentage field’s value does not exceed 40%, you might use a one-line script like:

```groovy
return CommissionPercentage < 0.40
```

In fact, this one-liner can be conveniently shortened by dropping the return keyword since the return keyword is always implied on the last line of a script:

```groovy
CommissionPercentage < 0.40
```

For slightly more complicated logic, your script might require some conditional handling. For example, suppose the maximum commission percentage is 40% if the salesperson’s job grade is less than or equal to 3, but 60% if the job grade is higher. Your script would grow a little to look like this:

```groovy
if (JobGrade <= 3) {
    return CommissionPercentage < 0.40
} else {
    return CommissionPercentage < 0.60
}
```

Scripts that you write for other purposes like complex validation rules or reusable functions may span multiple pages, depending on your needs.

When a context requiring a Groovy script will typically use a short (often, one-line) script, we emphasize that fact by calling it an expression. Technically the terms script and expression are interchangeable. Anywhere you can provide a one-line expression is also a valid context for providing a multi-line script if the need arises. Whether you provide a short expression or a multi-line script, the syntax and features at your disposal are the same. You need only pay attention that your code returns a value of the appropriate type for the context in which you use it.

The Groovy Scripting: Examples topic includes all the return types. This topic highlights the expected return type for each script example.

### Using Groovy Scripts in Your Application

There are a number of different contexts where you will use Groovy scripts as you modify existing objects or create new custom ones.

You will write shorter scripts to provide an expression to:

- Calculate a custom formula field’s value
- Calculate a custom field’s default value
- Make a custom field conditionally updatable, or
- Make a custom field conditionally required
- Define the condition for executing an object workflow

You will generally write somewhat longer scripts to define:

- A field-level validation rule
- An object-level validation rule
- A trigger to complement default processing
- Utility code in a global function, or
- Reusable behavior in an object function

If you anticipate calling the same code from multiple different contexts, any of your scripts can call the reusable code you write in either global functions or object functions. As their name implies, global functions can be called from scripts in any object or from other global functions. Object functions can be called by any scripts in the same object, or even triggered by a button in the user interface.

After exploring the Groovy basic techniques needed to understand the examples documented in Oracle Sales Cloud Groovy Scripting Reference at http://www.oracle.com/pls/topic/lookup?ctx=cloud132&id=CGSAC, see “Groovy Scripting: Examples” for a concrete example of each of these usages. Also see “Groovy Tips and Techniques” in Oracle Sales Cloud Groovy Scripting Reference at http://www.oracle.com/pls/topic/lookup?ctx=cloud132&id=CGSAC151 for getting the most out of Groovy in your application.

**Related Topics**

- Groovy Scripting: Examples

**Server Scripts: Explained**

Application Composer supports Groovy as the scripting language you use to enhance your applications. There are many different contexts in which you can use Groovy scripts. This topic illustrates the use of validation rules, triggers, and object functions, which you can define using the Server Scripts node for any standard or custom object. For a more detailed explanation of Groovy scripting using Application Composer, see the Oracle Sales Cloud Groovy Scripting Reference Guide.

The server scripts you can define for any standard or custom object include:

- **Validation rules**
  
  Write a script to validate either a field or an object.

- **Triggers**
  
  Write trigger "scripts" to automatically execute an action whenever a specific trigger "event" occurs.

- **Object functions**
  
  Write a function that can be reused in multiple contexts. For example, you can reuse an object function inside a trigger or validation rule.
Validation Rules

Validation rules are constraints that you can define on either a field or on an object. Write an expression or a longer script to validate a field or object before it can be saved. Define validation rules using the Server Scripts node for any standard or custom object.

- A field-level validation rule is a constraint you can define on any standard or custom field. The rule is evaluated at runtime whenever the corresponding field's value is set. When the rule executes, the field's value has not been assigned yet and your rule acts as a gatekeeper to its successful assignment.

For example, consider a custom TroubleTicket object with a Priority field. You can set a field-level validation rule to validate that the number entered is between 1 and 5.

The expression (or longer script) you write must return a boolean value that indicates whether the value is valid.

- If the rule returns true, then the field assignment will succeed so long as all other field-level rules on the same field also return true.
- If the rule returns false, then this prevents the field assignment from occurring, the invalid field is visually highlighted in the UI, and the configured error message is displayed to the end user. Since the assignment fails in this situation, the field retains its current value (possibly null, if the value was null before), however the UI component in the web page allows the user to see and correct their invalid entry to try again.


- An object-level validation rule is a constraint you can define on any standard or custom object. Use object-level rules to enforce conditions that depend on two or more fields in the object. This ensures that regardless of the order in which the user assigns the values, the rule will be consistently enforced. The rule is evaluated whenever the framework attempts to validate the object. This can occur, for example, upon submitting changes in a Web form, when navigating from one row to another, as well as when changes to an object are saved. (Rules are not evaluated if the user saves a record without making changes.)

For example, consider a TroubleTicket object with Priority and AssignedTo fields, where the latter is a dynamic choice list field referencing Contact objects whose Type field is a 'Staff Member'. You can set an object-level validation rule to validate that a trouble ticket of priority 1 or 2 can’t be saved without being assigned to a staff member.

The expression (or longer script) you write must return a boolean value that indicates whether the object is valid.

- If the rule returns true, then the object validation will succeed so long as all other object-level rules on the same object return true.
- If the rule returns false, then this prevents the object from being saved, and the configured error message is displayed to the end user.


Triggers

Triggers are scripts that you can write to complement the default processing logic for a standard or custom object. When a specific event occurs, triggers automatically execute an action that you specify in the trigger definition. You can define triggers both at the object level and at the field level, using the Server Scripts node for any standard or custom object. When you define a trigger, you select the specific event that will cause your script to automatically run. This specific event is also referred
to as a trigger. Oracle supplies a set number of these trigger "events" which you can pick from when defining your trigger "scripts."

- Field-level triggers are scripts that you write to execute an action in response to a change in another field’s value. When you define a trigger at the field level, you select the **After Field Changed** trigger and the field that this trigger is watching. You then define the action that you want to happen when the field’s value changes.

  The **After Field Changed** trigger calculates other derived field values when the value of the field that you specify changes. Do not use a field-level validation rule to achieve this purpose because while your field-level validation rule may succeed, other field-level validation rules may fail and stop the field’s value from actually being changed. Since generally you only want your field-change derivation logic to run when the field’s value actually changes, the After Field Changed trigger guarantees that you get this desired behavior.

  See "Defining a Field-Level Trigger to React to Value Changes" in the Oracle Sales Cloud Groovy Scripting Reference Guide.

- Similarly, object-level triggers are scripts that you write that execute an action once a specific event occurs. In the case of object-level triggers, you have many more trigger "events" to pick from, such as:
  - **After Create**
  - **Before Invalidate**
  - **Before Remove**
  - **Before Insert in Database**
  - **Before Update in Database**
  - **Before Delete in Database**
  - **Before Rollback in Database**
  - **After Changes Posted to Database**

  For example, consider a Contact object with a OpenTroubleTickets field that needs to be updated any time a trouble ticket is created. You can create a trigger on the TroubleTicket object using the **After Changes Posted to the Database** trigger event. When an event occurs, your trigger can automatically update the OpenTroubleTickets field with a new count.

  For a complete list of the trigger "events" that you can pick from, see "Defining an Object-Level Trigger to Complement Default Processing" in the Oracle Sales Cloud Groovy Scripting Reference Guide.

### Object Functions and Global Functions

You can write reusable code as either an object function or global function. Do this if you anticipate calling the same code from multiple different contexts. Object functions can be called by any script in the same object, or even triggered by a button in the user interface. Global functions can be called from scripts in any object or from other global functions.

- Object functions are useful for code that encapsulates business logic specific to a given object. You can call object functions by name from any other script related to the same object. In addition, you can call them using a button or link in the user interface.

  The supported return types and optional parameter types are the same as for global functions. For a list of the most common types for function return values and parameters, see “Defining Utility Code in a Global Function” in the Oracle Sales Cloud Groovy Scripting Reference Guide.
See also "Defining Reusable Behavior with an Object Function" in the Oracle Sales Cloud Groovy Scripting Reference Guide.

- Global functions are useful for code that multiple objects want to share. Write user-defined functions using Groovy scripts, which can be referenced in all Groovy script editors throughout Application Composer. For example, you could create two global functions to define standard helper routines to log the start of a block of Groovy script and to log a diagnostic message.

To call a global function, preface the function name with the adf.util. prefix. When defining a function, you specify a return value and can optionally specify one or more typed parameters that the caller will be required to pass in when invoked.

For a list of the most common types for function return values and parameters, see "Defining Utility Code in a Global Function" in the Oracle Sales Cloud Groovy Scripting Reference Guide.

Privileged Functions

When you define either an object function or global function, the function might run on an object where the runtime user has no privileges to create or update records. Allow users without access to an object’s data to run a function with full access, by doing two things:

1. While defining the function, check the Privileged check box to indicate that the function is privileged.
2. Confirm that the Privileged Script Administration role has the right level of access so that the object function can execute successfully. Access to objects is not given automatically. Instead, you must grant access using the Application Composer security UI.

At runtime, when a user invokes a privileged function from the UI, a temporary login session is activated with the privileged role, Privileged Script Administration. This privileged role has access to the object in your function, so no permission issues exist. The temporary login session lasts only for the duration of the single function call or anything that function calls internally.

For example, a sales representative has access to opportunity records only, not account records. When a sales representative edits an opportunity, there is a button that updates a related account using a privileged Groovy script. Even though the sales representative doesn’t have update privileges for the account object, when the sales representative clicks the button, the privileged Groovy script executes by switching to the privileged role context to complete the update to the account record.

To make this happen, use the Application Composer security UI to grant account access to the privileged role, Privileged Script Administration.

Global Functions: Explained

Global functions are useful for code that multiple objects want to share. You use global functions when you write Groovy scripts using the expression builder in Application Composer. Some global functions are delivered with Oracle Sales Cloud, ready for your use. Or, you can define new global functions.


This topic:

- Explains how to define new global functions.
• Provides a list of some global functions that are provided in Application Composer.

Defining Global Functions
To define a global function:

1. In Application Composer, navigate to the Common Setup pane, which displays in the regional area.
2. Click Global Functions.

**Note:** You must be in an active sandbox.

4. Specify the global function name and a return value.
5. Enter a description and example of the global function.
6. Optionally check the Privileged check box.

   If this function might run on an object where the runtime user has no privileges to create or update records, then check this box. Users without access to an object's data can still run this function at runtime, with full access. See "Server Scripts: Explained" for more details on privileged scripts.
7. Optionally specify one or more typed parameters that the caller will be required to pass in, when the function is invoked.
8. Specify the body of the function.
9. Validate and save your function.

Examples of Predefined Global Functions
This table lists the global functions that are provided for use in Application Composer.

**Note:** These global functions are not available for selection in the expression builder. Instead, to use these functions, manually type the function name into your script, prefacing the function name with the adf.util prefix.

<table>
<thead>
<tr>
<th>Global Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adf. util. getUserPartyId()</td>
<td>Returns the logged-in user's Party_ID.</td>
</tr>
<tr>
<td>adf. util. getUserPartnerCompanyId()</td>
<td>Returns the partner company's party_ID for the logged-in user, if the user is a partner user.</td>
</tr>
<tr>
<td>adf. util. getUserRootResourceOrgId()</td>
<td>Returns the organization_ID for the logged-in user's organization hierarchy root resource organization.</td>
</tr>
</tbody>
</table>

Accessing View Objects: Explained

A view object is an Oracle ADF component that simplifies querying and working with business object rows. You access view objects when you write Groovy scripts using the expression builder in Application Composer.

To access view objects in your scripts, use the newView() function for an object API name. The newView() function accesses a custom or standard view object, and creates a new view object instance that programatically accesses that business
object’s rows. For example, a common task that you will do with this new view object instance is to query some data. Do this by calling the findByKey() function on the view object to find a row by key.


This topic:

- Explains why the newView() function is useful in your scripts.
- Explains how to access view objects, either custom or standard, from the expression builder using the newView() function.
- Provides a list of the standard view objects that are provided in Application Composer.

newView() Function

When you write Groovy scripts in Application Composer, you’re usually in the context of a specific record from a specific object. For example, you can write a trigger script with a single line "setAttribute('Name','Acme Widgets Inc.' )" and the script will be executed on the user’s current record.

The newView() function, by contrast, lets you construct a new reference to an object which doesn’t require any contextual relationship to the current record. For example, the line "def myVO = newView('OpportunityVO')" produces an instance of the Opportunity view object that your script can query and read, and then add, delete, or update rows.

Accessing View Objects

To access view objects, use the newView() function for an object API name from within the expression builder in Application Composer:

1. Navigate to the expression builder from Application Composer.

   There are several ways to launch the expression builder in Application Composer. For example, launch the expression builder when editing a field to make it required.

2. In the expression builder palette, on the Functions tab, select the Other category and the newView() function.

3. Click Insert.

   A window displays that lists the view objects you can call in your script.
The objects don't have to be related to the current object to appear in this list.

Examples of Standard View Objects

The standard objects that are delivered with Oracle Sales Cloud provide view objects for use in your scripts. The previous section described how to access those view objects. This section provides some examples of standard view objects that are provided in application composer, and how you might use them in your scripts. Attributes that you would typically script against are also included.

For objects that are not extensible and thus not available in Application Composer, see the SOAP Web Services for Oracle Sales Cloud guide to view a list of attributes that you can script against.

<table>
<thead>
<tr>
<th>Standard View Object</th>
<th>Description</th>
<th>Typical Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Use this object to access the address for a given party in scripting, if the current object doesn’t have a view link to the Address object. Access this Address extensible object as a child of the Account, Contact, or Household objects.</td>
<td>Refer to the Address object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
</tbody>
</table>

<p>| CodeAssignment       | Use this object to access classifications assigned to a given party in scripting, if the current object doesn’t have a view link to this object. | Refer to the Trading Community Classification Code Assignment in the SOAP Web Services for Oracle Sales Cloud guide. |</p>
<table>
<thead>
<tr>
<th>Standard View Object</th>
<th>Description</th>
<th>Typical Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access this object as a child of the Account or Contact objects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CommonLookup</td>
<td>Access application common lookups in scripting.</td>
<td>LookupType, LookupCode, Tag, EnabledFlag, StartDateActive, EndDateActive, Meaning, Description</td>
</tr>
<tr>
<td>Contact</td>
<td>Use this object to access customer contact information in scripting, if the current object doesn't have a view link to this object.</td>
<td>Refer to the Customer Contact Profile object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
<tr>
<td>FndTreeVersion</td>
<td>Use this object in scripting to access tree versions.</td>
<td>TreeStructureCode, TreeCode, TreeVersionID, Status, EffectiveStartDate, EffectiveEndDate, TreeVersionName</td>
</tr>
<tr>
<td>Location</td>
<td>Use this object to update or create physical location fields.</td>
<td>Refer to the Trading Community Location SDO in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
<tr>
<td>OrganizationParty</td>
<td>Use this object to get the organization party and all of its children when you have the organization PartyID in your script, and you don't have a view link from the current object to the Account object.</td>
<td>Refer to the Trading Community Organization Details in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
<tr>
<td>OrganizationProfile</td>
<td>Access this Account extensible object as a child of an OrganizationParty row or directly get the profile if you have a PartyID.</td>
<td>Refer to the Account object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
<tr>
<td>OriginalSystemReference</td>
<td>Use this object to get the ID for given TCA object based on the source system and source system reference information.</td>
<td>Refer to the Trading Community Original System Reference in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
</tbody>
</table>
Standard View Object | Description | Typical Attributes
--- | --- | ---
PersonParty | Use this object to get the Person Party and all of its children when you have the person PartyID in your script, and you don’t have a view link from the current object to Account object. | Refer to the Trading Community Person Details in the SOAP Web Services for Oracle Sales Cloud guide..

PersonProfile | Access this Contact extensible object as a child of a PersonParty row or directly get the profile if you have a PartyID. | Refer to the Contact object in Application Composer, and review the descriptions provided for all attributes.

Relationship | Use this object in scripting if you have a RelationshipID on the current object and that object doesn’t have a view link to this object. Access this Relationship extensible object as a child of the Account, Contact, or Household objects. | Refer to the Relationship object in Application Composer, and review the descriptions provided for all attributes.

Resource | Use this Resource extensible object in scripting to get the resource object details if you have a user or resource PartyID, and the current object ID doesn’t expose a view link to this object. | Refer to the Trading Community Resource Profile in the SOAP Web Services for Oracle Sales Cloud guide..

Related Topics
- SOAP Web Services for Oracle Sales Cloud

Supported Classes and Methods: Explained

Groovy is a standard, dynamic scripting language for the Java platform for which Application Composer provides support. This topic covers the supported classes and methods for use in Groovy scripts.

Classes and Methods

When writing Groovy scripts, you may only use the classes and methods that are documented in the following table. Using any other class or method may work initially, but will throw a run time exception when you migrate your code to later versions. Therefore, we strongly suggest that you ensure the Groovy code you write adheres to the classes and methods shown here.

For each class, in addition to the method names listed in the table, the following method names are also allowed:

- equals()
- hashCode()
- toString()

In contrast, the following methods are never allowed on any object:

- finalize()
- getClass()
- getMetaClass()
- notify()
- notifyAll()
- wait()

**Note:** The following supported classes and methods will expand over time depending on customer requirements and business need. Thus, periodically review this table to assess what is newly supported in each release.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADFContext</td>
<td>getLocale()</td>
<td>oracle.adf.share</td>
</tr>
<tr>
<td></td>
<td>getSecurityContext()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUserRoles()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isUserInRole()</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>getArray()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>getElemType()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getList()</td>
<td></td>
</tr>
<tr>
<td>ArrayList</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Arrays</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>AttributeDef</td>
<td>getAttributeKind()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getIndex()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getJavaType()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPrecision()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getScale()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUIHelper()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUpdateableFlag()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isMandatory()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>isQueriable()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AttributeHints</td>
<td>getControlType()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getDisplayHeight()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayHint()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayWidth()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormat()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormattedAttribute()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormatter()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormatterClassName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getHint()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLocaleName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parseFormattedAttribute()</td>
<td></td>
</tr>
<tr>
<td>AttributeList</td>
<td>getAttribute()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getAttributeIndexOf()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAttributeNames()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setAttribute()</td>
<td></td>
</tr>
<tr>
<td>BaseLobDomain</td>
<td>closeCharacterStream()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>closeInputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>closeOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getInputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLength()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getCharacterStream()</td>
<td></td>
</tr>
<tr>
<td>BigDecimal</td>
<td>Any constructor</td>
<td>java.math</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BigInteger</td>
<td>Any constructor</td>
<td>java.math</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BitSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Blob</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BlobDomain</td>
<td>Any constructor</td>
<td>oracle, jbo.domain</td>
</tr>
<tr>
<td></td>
<td>getBinaryOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBinaryStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBufferSize()</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Char</td>
<td>Any constructor</td>
<td>oracle, jbo.domain</td>
</tr>
<tr>
<td></td>
<td>bigDecimalValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bigIntegerValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>booleanValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>doubleValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>floatValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>intValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>longValue()</td>
<td></td>
</tr>
<tr>
<td>Clob</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ClobDomain</td>
<td>Any constructor</td>
<td>oracle, jbo.domain</td>
</tr>
<tr>
<td></td>
<td>toCharArray()</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Collections</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Comparator</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>DBSequence</td>
<td>Any constructor</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Any constructor</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>compareTo()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dateValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stringValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>timestampValue()</td>
<td></td>
</tr>
<tr>
<td>Dictionary</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Enum</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>EnumMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>EnumSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Enumeration</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>EventListener</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>EventListenerProxy</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>EventObject</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Exception</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ExprValueErrorHandler</td>
<td>addAttribute(), clearAttributes(), raise(), raiseLater(), warn()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Float</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Formattable</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>FormattableFlags</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Formatter</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>GregorianCalendar</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>HashMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>HashSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Hash table</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>IdentityHashMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Iterator</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>JboException</td>
<td>getDetails()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getErrorCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getErrorParameters()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLocalizedMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProductCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td>JboWarning</td>
<td>Any constructor</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getDetails()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getErrorCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getErrorParameters()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLocalizedMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProductCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>toStringFormat()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LinkedHashMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>LinkedHashMapSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>LinkedList</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>List</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ListIterator</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ListResourceBundle</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Locale</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Long</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Map</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Math</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>MathContext</td>
<td>Any constructor</td>
<td>java.math</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>NClob</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>NameValuePairs</td>
<td>Any constructor</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getAttribute()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>NativeTypeDomainInterface</td>
<td>getNativeObject()</td>
<td>oracle, jbo.domain</td>
</tr>
<tr>
<td>Number</td>
<td>Any constructor</td>
<td>oracle, jbo.domain</td>
</tr>
<tr>
<td></td>
<td>BigDecimalValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BigIntegerValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>booleanValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>byteValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>doubleValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>floatValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>intValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>longValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shortValue()</td>
<td></td>
</tr>
<tr>
<td>Observable</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Observer</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>PriorityQueue</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Properties</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>PropertyPermission</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>PropertyResourceBundle</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Queue</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Random</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>RandomAccess</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>Ref</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td>ResourceBundle</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>Row</td>
<td>getAttribute(), getAttributeHints(), getKey(), getLookupDescription(), getOriginalAttributeValue(), getPrimaryRowState(), getSelectedListDisplayValue(), getSelectedListDisplayValues(), isAttributeChanged(), isAttributeUpdateable(), remove(), revertRow(), revertRowAndContainers(), setAttribute(), validate()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>RowId</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td>RowIterator</td>
<td>createAndInitRow(), createRow(), findByPrimaryKey()</td>
<td>oracle.jbo</td>
</tr>
</tbody>
</table>

---

---

---
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowSet</td>
<td>avg()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>count()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>createAndInitRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>createRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>executeQuery()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findByKey()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findRowsMatchingCriteria()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>first()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAllRowsInRange()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getCurrentRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getEstimatedRowCount()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasNext()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasPrevious()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>insertRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>last()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>next()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>previous()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>reset()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scanner</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>SecurityContext</td>
<td>getUserUserName()</td>
<td>oracle.adf.share.security</td>
</tr>
<tr>
<td></td>
<td>getUserProfile()</td>
<td></td>
</tr>
<tr>
<td>Session</td>
<td>getLocale()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getLocaleContext()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUserData()</td>
<td></td>
</tr>
<tr>
<td>Set</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>SimpleTimeZone</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>SortedMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>SortedSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Stack</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StackTraceElement</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StrictMath</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>String</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringBuffer</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringBuilder</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringTokenizer</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Struct</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAttribute()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>setAttribute()</td>
<td></td>
</tr>
<tr>
<td>StructureDef</td>
<td>findAttributeDef()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getAttributeIndexDef()</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>TimeZone</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Timer</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>TimerTask</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Time stamp</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Time stamp</td>
<td>Any constructor</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>compareTo()</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 7
Using Groovy Scripts

Oracle Sales Cloud
Extending Sales
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dateValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stringValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>timestampValue()</td>
<td></td>
</tr>
<tr>
<td>TreeMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>TreeSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>UUID</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>UserProfile</td>
<td>getBusinessCity()</td>
<td>oracle. adf. share. security. identitymanagement</td>
</tr>
<tr>
<td></td>
<td>getBusinessCountry()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessEmail()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessFax()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessMobile()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPOBox()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPager()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPhone()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPostalAddr()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPostalCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessState()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessStreet()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDateofBirth()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDateofHire()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDefaultGroup()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDepartment()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDepartmentNumber()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDescription()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayName()</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Some of these methods may return null if the corresponding attribute of the user record is not populated in the identity store or if the identity provider does not support those methods.
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEmployeeNumber()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getEmployeeType()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getFirstName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getGUID()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getGivenName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getHomeAddress()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getHomePhone()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getInitials()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getJpegPhoto()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getLastName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getMaidenName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getManager()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getMiddleName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getNameSuffix()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getOrganization()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getOrganizationalUnit()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPreferredLanguage()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPrincipal()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getProperties()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getProperty()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getTimeZone()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getTitle()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getUIAccessMode()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getUniqueName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getUserID()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getUserName()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getWirelessAcctNumber()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getDetails()</td>
<td></td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>getErrorCode()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getErrorParameters()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>getLocalizedMessage()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getMessage()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getProductCode()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getProperty()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ViewCriteria</td>
<td>createAndInitRow()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>createRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>createViewCriteriaRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findByKey()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findRowsMatchingCriteria()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>first()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAllRowsInRange()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getCurrentRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getEstimatedRowCount()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasNext()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasPrevious()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>insertRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>last()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>next()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>previous()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset()</td>
<td></td>
</tr>
<tr>
<td>ViewCriteriaItem</td>
<td>getValue()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>makeCompound()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setOperator()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setUpperColumns()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setValue()</td>
<td></td>
</tr>
<tr>
<td>ViewCriteriaItemCompound</td>
<td>ensureItem()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>makeCompound()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>setOperator()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setUpperColumns()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setValue()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ensureCriteriaItem()</td>
<td></td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>getConjunction()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isUpperColumns()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setConjunction()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setUpperColumns()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>appendViewCriteria()</td>
<td></td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>avg()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>count()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createAndInitRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createViewCriteria()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>executeQuery()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>findByKey()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>findRowsMatchingCriteria()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>first()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getAllRowsInRange()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getCurrentRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getEstimatedRowCount()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getMaxFetchSize()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hasNext()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hasPrevious()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>insertRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>last()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>min()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>previous()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using Groovy Scripting: Examples

This topic contains examples of application changes that you can perform using Groovy scripts. Application Composer leverages Groovy to enable you to enhance your application changes. Groovy is a standard, dynamic scripting language for the Java platform for which the Application Composer provides support.

The following examples are covered in this topic:

- Using println function
- Making fields conditionally required
- Making fields conditionally updatable
- Adding validation to fields

> Note: It is assumed that a custom object called Help Request exists and is available on the Navigator menu.

Using println Function

In this example, you add a `println` function to a trigger to view an opportunity's Win Probability. Whenever the Win Probability field is updated, the `println` function in the trigger performs an update, and you can see the output in the Run Time Messages user interface (UI) within the Application Composer.

To view an opportunity's WinProbability:

1. Navigate to Application Composer.
2. Expand Standard Objects and then expand Opportunity.
5. In the Trigger field, select Before Update in Database.
6. In the **Trigger Name** field, enter **TestPrintln**.
7. Under the **Trigger Definition** region, enter the following script in the expression text box:
   
   ```groovy
   println("Before Update Trigger. The new value of the Win Probability is" + nvl(WinProb, "Win Probability was null").
   
   nvl() ensures that the variables are null-aware.
8. Click the **Validate** icon. Confirmation appears when the script is parsed successfully.
9. Click **OK**.
10. In the **Navigator** menu, click **Opportunities**.
11. Click **Create Opportunity** in the left pane.
12. In the **Name** field, enter **Opportunity Trigger Test**.
13. In the **Win Probability (%)** field, enter **50**.
14. Click the **Save and Close** button.
15. On the **Overview** page, select the **Opportunities** tab.
16. Click the **Opportunity Trigger Test** link in the table.
17. In the **Win Probability (%)** field, enter **25**.
18. Click the **Save and Close** button.
19. Navigate to Application Composer.
20. On the **Overview** page, click **Run Time Messages**.
21. Click the **Get Latest Log Messages** button.
22. Locate the message that you wrote in your `println` function.

**Tip:** Sort the messages in descending order to locate quickly.

### Making Fields Conditionally Required

In this example, you add the following two custom fields to the Help Request object. You make one of them conditionally required based on the value you select in the other field.

1. **Priority** field of type Fixed Choice List.
2. **Justification** field of type text that is conditionally set to required depending on the value in the **Priority** field.
   - If the value in the Priority field is set to Urgent, then the Justification field appears as a mandatory or required field.
   - Else, the Justification field remains optional.

To create fields in application composer and make one of them conditionally required:

1. Navigate to Application Composer.
2. Expand **Custom Objects**.
3. Expand **Help Request** and click **Fields**.
4. Under Custom tab, select **Action > Create**.
5. In the Select Field Type window, select **Text** and click **OK**.
6. On the Create Text Field page, enter **Justification** in the **Display Label**.
7. Click **Save and Close**.
8. On the Fields page, select **Action > Create** under the Custom tab.
9. In the Select Field Type window, select **Choice List (Fixed)** and click **OK**.
10. On Create Fixed Choice List page, enter **Priority** in the **Display Label**.
11. Under the List of Values region, click the **Create a New Lookup Type** icon.
12. In Create Lookup Type dialog, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>Help Request Priority</td>
</tr>
<tr>
<td>Lookup Type</td>
<td>HR_PRIORITY</td>
</tr>
</tbody>
</table>

13. Select Action > Create, and specify the following in the first row:

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>Urgent</td>
</tr>
<tr>
<td>Lookup Code</td>
<td>URG</td>
</tr>
<tr>
<td>Display Sequence</td>
<td>1</td>
</tr>
</tbody>
</table>

14. Select Action > Create, and specify the following in the second row:

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>Important</td>
</tr>
<tr>
<td>Lookup Code</td>
<td>IMP</td>
</tr>
<tr>
<td>Display Sequence</td>
<td>2</td>
</tr>
</tbody>
</table>

15. Click Save.

16. On Create Fixed Choice List page, select the Fixed Value option in the Default Value region, and then select Important as the default value.

17. Click the Save and Close button.

18. Under the Custom tab, click the Justification link.

19. Under Constraints region, select Required check box and click the expression builder icon next to it.

20. Enter the following into the (script) text box:

   ```java
   try { if(nvl(Priority_c,"") == "URGENT") { return true } else { return false } } catch(e) {
   println("Error with the Required property of the Justification field in the Help Request object")
   }
   ```

21. Click the OK button.

22. Under Constraints region, select Priority from the Depends On list.

23. Click the Save and Close button.

24. Under Help Request in the left pane, click Pages.
25. Click the **Edit Summary Form** link.
   Edit Details Page Summary Form page opens.
26. Under Configure Default Summary region, move the **Priority** and **Justification** fields to the **Selected Fields** box.
   Ensure that **Priority** field is above the **Justification** field. If it’s not, use the up or down arrow button on the right to adjust the sequence.
27. Click the **Save and Close** button.
28. In the **Navigator** menu, click **Help Request**.
29. Click any active Help Request in the list to open its edit page.
30. Select **Important** from the **Priority** list.
31. Click the **Save** button.
   The help request has been saved.
32. Now select **Urgent** in the **Priority** list.
   An asterisk appears next to the **Justification** field that indicates a required field.
33. Click the **Save** button.
   An error message appears, because you did not enter a value in the **Justification** field before saving.
34. Enter **Laptop is on fire** in the **Justification** field.
35. Click the **Save and Close** button.
   The help request is now saved.

### Making Fields Conditionally Updatable

In this example, you add the following two custom fields to your Help Request object and then enter a script to make one of them conditionally updatable.

1. **Executive Sponsor Program** check box
2. **Executive Sponsor** text field

If the Executive Sponsor Program check box is selected, the Executive Sponsor field can be updated. Else, the Executive Sponsor field is disabled.

To create these fields and make one of them conditionally updatable:

1. Navigate to Application Composer.
2. Expand **Custom Objects**.
3. Expand **Help Request** and click **Fields**.
4. Under Custom tab, select **Action > Create**.
5. In the Select Field Type window, select **Text** and click **OK**.
6. On the Create Text Field page, enter **Executive Sponsor** in the **Display Label**.
7. Click **Save and Close**.
8. On the Fields page, select **Action > Create** under the Custom tab.
9. In the Select Field Type window, select **Check box** and click **OK**.
10. On the Create Text Field page, enter **Executive Sponsor Program** in the **Display Label**.
11. Click **Save and Close**.
12. On the Fields page, click the **Executive Sponsor** link.
   Edit Check box Field: Executive Sponsor page opens.
13. Under Constraints region, select the **Updatable** check box and click the expression builder icon next to it.
Expression builder dialog opens.

14. Enter the following into the (script) text box:
   ```java
   try{ if(nvl(ExecutiveSponsorProgram_c,"") == "N") { return false } else { return true } } catch(e) {
   println("Error with the Updatable property of Executive Sponsor field in the Help Request object") }
   ```

15. Click the OK button.

16. Under Constraints region, select Executive Sponsor Program from the Depends On list.

17. Click the Save and Close button.

18. Under Help Request in the left pane, click Pages.

19. Click the Edit Summary Form link.

20. Under Configure Default Summary region, move the Executive Sponsor and Executive Sponsor Program fields to the Selected Fields box.

   Ensure that Executive Sponsor Program field is above the Executive Sponsor field. If it’s not, use the up or down arrow button on the right to adjust the sequence.

21. Click the Save and Close button.

22. In the Navigator menu, click Help Request.

23. Click any active Help Request in the list to open its edit page.

24. Select the Executive Sponsor Program check box, and click the Save button.

   At this point, you can enter a name into the Executive Sponsor field.

25. Now clear the selection in the Executive Sponsor Program check box, and then click the Save button.

   You can’t enter anything in the Executive Sponsor field, because you have not selected the Executive Sponsor Program check box.

## Adding Validations to Fields

In this example, you add a validation to a field using an expression. You first add a custom text field called Corporate E-Mail to your Help Request object, and then you add a validation to check the syntax of the e-mail address.

To create a field and add validation:

1. Navigate to Application Composer.
2. Expand Custom Objects.
3. Expand Help Request and click Fields.
4. Under Custom tab, select Action > Create.
5. In the Select Field Type window, select Text and click OK.
6. On the Create Text Field page, enter Corporate E-Mail in the Display Label.
7. Click Save and Close.
8. In the left pane, click Pages under Help Request.

   Help Request: Pages opens.


   Edit Creation Page opens.

10. Under Configure Creation page region, move Corporate E-Mail to the Selected Fields box.
11. Click the **Save and Close** button.
12. Under Details Page region, click **Edit Summary Form**. Edit Details Page Summary Form page opens.
13. In the Configure Default Summary region, move the **Corporate E-Mail** to the **Selected Fields** box.
14. Click the **Save and Close** button.
15. In the left pane, click **Server Scripts** under **Help Request**.
16. Under Validation Rules tab, select **Action > Add** under the Field Rules region.
17. Specify the following values under Create Field Validation Rule page:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Corporate E-Mail</td>
</tr>
<tr>
<td>Rule Name</td>
<td>CheckValid</td>
</tr>
<tr>
<td>Error Message</td>
<td>Invalid syntax in Corporate E-Mail. Enter a valid syntax for e-mail address</td>
</tr>
</tbody>
</table>

18. In the script textbox, enter:
```
newValue == null || newValue =~ /[A-Za-z0-9-]+\.[A-Za-z0-9-]+\*[A-Za-z0-9-]+\*\.[A-Za-z]{2,}/
```
19. Click the **Save and Close** button.
20. In the Navigator menu, click **Help Request**.
21. Click any active Help Request in the list to open its edit page.
22. Enter `mhoope.oracle.com` in the **Corporate E-Mail** field.
23. Click the **Save** button.
   The error message that you had specified appears, because the e-mail address does not have proper syntax.
24. Enter `mhoope@oracle.com` in the **Corporate E-Mail** field.
25. Click the **Save** button.
   You are now able to save the Help Request after entering a valid syntax for e-mail address.

---

**Calling REST Web Services from Groovy Scripts**

**Calling REST Web Services: Explained**

You can call RESTful web services from your Groovy scripts in Application Composer. You might call a web service for access to internal or external data, or to perform a calculation on your data.

Calling RESTful web service methods from your scripts involves two high-level steps:

1. Creating a reference to the web service.
   This includes registering the web service’s endpoint with a variable name that you use in your Groovy script.
2. Writing a Groovy script in the Expression Builder that calls the web service.
   For each call, the script must prepare the inbound arguments to the web service, call a web service method, and then process the return value from the web service.
Creating a Web Service Reference

To register a web service for use in your scripts, you first select Web Services in the Common Setup pane in Application Composer. You can select either REST or SOAP. To register REST services, select REST. You then associate a web service variable name with a URL that provides the location of the resource that represents the service you want to call.

For example, you might register a web service variable name of TwitterSearch for a web service that your application needs to call for retrieving tweets from Twitter, in this case, about Yosemite. The URL for this web service’s location might be:

https://api.twitter.com/1.1/search/tweets.json?q=yosemite

Or, you might want to use this feature with an internal web service, such as within Oracle Sales Cloud. In this case, the URL might be:

http://host:port/OpptyService/rest/v1/Oppty?q=OpptyId=##OpptyId##

Of course, the server name, the port number, and path name for your actual service will be different. If the port number is omitted, then it is assumed that the service is listening on the default HTTP port number 80.

Writing a Groovy Script to Call a Web Service

When you call a web service from a Groovy script, the script must prepare the arguments to the web service before calling a web service method, and then process the data returned from the web service. Your script can also pass a structured payload to and from a web service.

You insert the code for the call to the web service from the Web Services tab in Expression Builder. The Web Services list displays the set of registered web service variable names and the Functions list displays the available methods for a given web service.

To insert a call to a RESTful web service in a Groovy script:

1. Select the Web Services tab in Expression Builder.

2. Select REST.

3. Select a variable name from the Web Services list.

4. Select a method from the Functions list.

   The code that will be inserted is shown under Function Signature.
The information under Function Signature includes the parameter types and also the return type to indicate the type of variable the result of the call should be assigned to. The possible return types are:

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>Void</td>
</tr>
<tr>
<td>Scalar values (integer, string and so on)</td>
<td>The actual Java return type</td>
</tr>
<tr>
<td>Object</td>
<td>Map</td>
</tr>
<tr>
<td>Collection</td>
<td>List</td>
</tr>
</tbody>
</table>

5. Position the cursor at the place in the script where you want to insert the web service call.
6. Click the **Insert** button to insert the code to call the web service method.

A web service call from a Groovy script has the following syntax:

```
adf.webServices.YourServiceVariableName.MethodName(args)
```

For example:

```
adf.webServices.ContactAddressAPI.Get("7627")
```

7. Click **Submit**.

### Registering REST Endpoints: Explained

In the Groovy scripts that you use in Application Composer, you can include calls to both internal and external web services. For each web service that you call in your scripts, you must first register the REST endpoint that you want to access. To register a REST endpoint, you specify the file location and the security scheme, if any, used to access the web service. Registering the endpoint creates a web service reference that you can use in your Groovy scripts.

#### Creating a Web Service Reference

To create a web service reference, do the following in Application Composer:

1. Confirm that you are in an active sandbox session.
2. In Application Composer, under Common Setup, click **Web Services**.
3. Click the **Create Web Service Reference** icon.
4. Select **REST**, then **OK**.
5. Enter the name for this web service reference.

   This name is simply an identifier that is used in the list of web services on the Expression Builder’s Web Services tab.
6. Specify the URL of the file location for the web service that you want to integrate with.
7. If you select an authentication scheme, then specify the required information. For secure communication with a web service, you can use various schemes for authenticating user credentials and ensuring security. The following schemes are supported for web services from Groovy scripts:
   - None
Select this option to specify that no security scheme is used.
- Call with basic authentication
- Propagate user identity using SAML over SSL
- Propagate user identity using SAML
- Call using OAUTH

Do not use this scheme with non-Oracle web service endpoints. This scheme currently supports only resources protected with Oracle Cloud OAuth server.

For the security schemes that require user name and password credentials, specify a credential key. The web service provider will tell you about the credentials that you must use for a particular web service.

8. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder.

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Type</td>
<td>Select the check box for the method type you want to expose in the Expression Builder.</td>
</tr>
<tr>
<td></td>
<td>- GET</td>
</tr>
<tr>
<td></td>
<td>- PUT</td>
</tr>
<tr>
<td></td>
<td>- POST</td>
</tr>
<tr>
<td></td>
<td>- PATCH</td>
</tr>
<tr>
<td></td>
<td>- DELETE</td>
</tr>
</tbody>
</table>

For each method you want to expose, specify the following information.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>The name of the method you selected appears here. By default it’s the same as the method type (such as POST) but you can change it.</th>
</tr>
</thead>
</table>
Selection | Description
--- | ---
Format | Select a format for the method, based on what the selected web service returns.
  - XML
  - JSON

This information is provided by your web service provider or web service documentation.

Request Payload
Specify the object structure of the payload, if needed. You can do this in one of two ways:
  - Directly provide the schema URL that represents the object structure.
  - Provide a code sample in JSON or XML format. This is an optional parameter useful for displaying reference hints in the Expression Builder.

To obtain a JSON code sample, execute the API externally using a REST client, such as Postman. The request payload in your Postman execution forms the Request Payload for this Code Sample parameter.

This section is optional.

Response Payload
If the method will return a response (for example, GET), specify the response object structure in which you want the response payload to be returned:
  - Schema URL
    - A URL that provides a structure for the data but doesn’t include any values.
  - Code Sample
    - If you don’t have a schema available, you can select this option and paste a sample response (for example, from the service documentation) in JSON or XML format into this field. This parameter is useful for displaying reference hints in the Expression Builder.
    - To obtain a JSON code sample, execute the API externally using a REST client, such as Postman. The response payload in your Postman execution forms the Response Payload for this Code Sample parameter.

![Image of Postman request](image)

**Note:** You must include an entry in the Response Payload field. If you do not, then Groovy returns "null" instead of the response payload. If you don’t want to include
You can always edit existing web service references, for example, to change the security scheme used or the settings used for a particular security scheme.

After you create a web service reference, the name of the web service appears in the list available on the Web Services tab in the Expression Builder. When you select a web service from the list, you can then select any of the functions provided by the web service for use in your Groovy scripts.

Tip: When managing web service references, click the Refresh icon in the Web Services page to make sure the list is up to date. Read "Refreshing the List of Web Service References" below for information about when you need to click Refresh.

Refreshing the List of Web Service References

If new methods are added for a web service, you must click the Refresh icon on the Web Services page so that the web service reference is updated. Otherwise, the new methods will not be available for the web service in the Expression Builder.

The Refresh icon is applicable whenever the service contract with the client changes. This can result in new methods, in the changing of the signature of existing methods, or in the deletion of existing methods.

You might also want to click Refresh to display any new web service references that have been created in a separate user session.

Configuring Security

Configure security differently, depending on whether you are creating a web service reference to an associated service endpoint or unassociated service endpoint.

When creating a web service reference to an associated service endpoint, such as to Oracle Java Cloud Service - SaaS Extension residing in the same identity domain, the required setup including SSO enablement is completed during association.

- To synchronize users between Oracle Sales Cloud (SaaS) and Oracle PaaS (JCS-SaaS Extension), review the steps in Synchronizing Oracle Sales Cloud, Oracle HCM Cloud, and Oracle ERP Cloud User Identities and Roles to SIM at https://cloud.oracle.com/developer/solutions?tab=tasks&solutionguid=OCPSI-GUID-C3F3348C-8C74-4F17-B9CE-0893CE2FA7CC.

When creating a web service reference to an unassociated service endpoint, most likely a non-Oracle Cloud service, do the following setup:

- In this case, you must create a service request for your administrator.
- You must retrieve the server’s CA SSL certificate from the service provider and attach it in the service request along with the location, and error details.
- If basic authentication is not necessary, then use the SAML over SSL authentication scheme.

The administrator will import the server SSL certificate into the tenant domain and inform you when this has happened.

Resolving Security Setup Errors

If some security setup has not been performed, then you might receive some errors when the web service is called from a Groovy script.

- A bad encryption error, when message protection is used
- A PolicyEnforcementException error when message protection security is used.

You must create a service request for your administrator to resolve the errors. Retrieve the server’s encryption certificate and the issuer certificate from the service provider and attach them both in the service request along with the location and the error details.

Moving Application Changes

You can download the application changes you make in a "source" environment, and upload them into a "target" environment. This can save you time when working with application changes across multiple environments.

To do this, do the following:

1. Use the Configuration Set Migration page to create a set of all changes and extensions made to an application environment.
2. Then, download the configuration set and upload it into another environment.

This is often referred to as configuration set migration, or CSM.

However, web service references created in Application Composer in the source environment won’t work in your target environment after the migration. Therefore, after you upload the configuration set to the target environment, you must re-create the web service references using Application Composer in the target environment, as well.

Integrating with Oracle SaaS REST Services: Explained

You can make REST outbound calls from Sales Cloud to an ADF-based REST endpoint within the same Sales Cloud instance. This is useful when there is a need to do cross-application calls to fetch data from objects that might not be accessible using Groovy. This topic illustrates different ways of making a call to an ADF-based REST endpoint within the same Sales Cloud instance.

In this topic, you will learn how to:

1. Register a base URL for an internal REST endpoint.
2. Append to that base URL in your Groovy scripts so that you can make queries to various objects.
4. Use the finders that are provided with Oracle Sales Cloud.
5. Use additional parameters to modify a REST endpoint.
6. Make query parameter calls.
7. Create a POST request to create a new contact address.
Registering the Base URL

To integrate with a REST endpoint, you must first register it as a variable in Application Composer so that you can later include that variable in your Groovy scripts. When you register the REST endpoint, you don’t have to specify the entire endpoint. Instead, to save time, you can register only the base URL so that the REST endpoint is reusable. Later, in your Groovy script, you can specify the rest of the endpoint. For example, in your Groovy script, you can reference the base URL and then specify if you are making a call to retrieve information about contacts or accounts.

Let’s look at an example. In this example, you will register a base URL. In the next section, you will use the base URL in a Groovy script.

First, register the base URL:

1. In Application Composer, under Common Setup, click Web Services.
2. Click the Create Web Service Reference icon.
3. Select REST, then OK.
4. Enter the variable name for this reference. For example, GetObjects.
5. Enter the base URL for the REST endpoint that you want to integrate with. For example:

   https://host:port/crmCommonApi/resources/latest/##Object##

6. Use basic authentication. Select Call with basic authentication.
7. In the Credential Key field, specify a name for the secret key that can be used to access the web service. This key name along with the user name and password is stored in the credential store.
8. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a GET method.

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>GET</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Schema URL</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
<tr>
<td>Code Sample</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;items&quot; : [ ],</td>
</tr>
<tr>
<td></td>
<td>&quot;count&quot; : 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;hasMore&quot; : false,</td>
</tr>
<tr>
<td></td>
<td>&quot;limit&quot; : 25,</td>
</tr>
<tr>
<td></td>
<td>&quot;offset&quot; : 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;links&quot; : [ {</td>
</tr>
<tr>
<td></td>
<td>&quot;rel&quot; : &quot;self&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;href&quot; : &quot;<a href="http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c">http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot; : &quot;T1_c&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;kind&quot; : &quot;collection&quot;</td>
</tr>
<tr>
<td></td>
<td>} ]</td>
</tr>
</tbody>
</table>
```

Finally, use the base URL in a Groovy script. Read the next section to learn how.
Appending to the Base URL

After you register the base URL, you can then reference the base URL in your Groovy scripts and manipulate the REST endpoint to specify which object you want to integrate with. For example, you can reference the same base URL to query either contacts or accounts. To do this, further append to the URL to specify which object you want to integrate with.

Continuing the example from the previous section, let’s now reference the REST endpoint in a Groovy script.

```groovy
def conn = adf.webServices.GETObjects
try{
def Object = "contacts"
def result = conn.GET(Object)
println("Headers:"+conn.responseHTTPHeaders)
println("Status:"+conn.statusCode)
println("Contacts:"+contacts)

}catch(Exception e){
println("Headers:"+conn.responseHTTPHeaders)
println("Status:"+conn.statusCode)
println("Output"+conn.httpErrorResponse)
}
```

To query accounts using the same base URL, use this Groovy script:

```groovy
def conn = adf.webServices.GetObjects
try{
def Object = "accounts"
def accounts = conn.GET(Object)
println("Headers:"+conn.responseHTTPHeaders)
println("Status:"+conn.statusCode)
println("Accounts:"+accounts)

}catch(Exception e){
println("Headers:"+conn.responseHTTPHeaders)
println("Status:"+conn.statusCode)
println("Output"+conn.httpErrorResponse)
}
```

Dynamically Passing Parameters in Your Groovy

When integrating with a REST endpoint, you can pass a parameter to refine your query. For example, when querying contacts, you might want to retrieve information for a particular party ID. You can do this without hard coding the party ID by dynamically passing the party ID in your Groovy script.

You can dynamically pass parameters based on where your Groovy is being called from. The page where the script is called from has context which you can use. For example, the page has context about the logged-in user, or about the contact record that the user is viewing. When you provide the URL, you can specify the party ID as a URL parameter.

Let’s say that you defined a custom attribute to hold contact address information in a denormalized (concatenated) form. You require the denormalized form of the address to sync contact information into one of your legacy systems. You can achieve this use case by querying the Sales Cloud Contacts API to fetch the address using Party Number as the parameter.

In this case, the REST endpoint would be:

```
https://<host:port>/crmCommonApi/resources/latest/contacts/##PartyNumber##/child/Address
```

And your Groovy script could look like this:

```groovy
def ContactAddressAPI = adf.webServices.ContactAddressAPI
try
{
def contactAddress = ContactAddressAPI.GET("<PartyNumber>")
```
```groovy
def address1 = contactAddress.items[0].Address1
def address2 = contactAddress.items[0].Address2
def city = contactAddress.items[0].City
def state = contactAddress.items[0].State
def country = contactAddress.items[0].Country
def postalCode = contactAddress.items[0].PostalCode

def denormalizedAddress = address1 + "", address2 + ",", city + ",", state + ",", country + ",", postalCode
return denormalizedAddress // concatenated address
}
catch(Exception ex)
{
    println(ContactAddressAPI.statusCode+""); // for diagnostic logging
    println(ContactAddressAPI.httpErrorResponse+""); // for diagnostic logging
    throw ex;
}
```

**Using Finders**

In your Groovy scripts, you can further refine the records that you retrieve by using finders to search a collection of data. Finders are predefined with Oracle Sales Cloud and are similar to a saved search. In your script, you state the finder name and include corresponding finder variables, if any, depending on the finder that you're using.

As mentioned earlier, you can attach the PartyNumber directly to the REST endpoint URL itself to retrieve a specific contact. For example:

```
https://<host:port>/crmCommonApi/resources/11.12.1.0/contacts/##PartyNumber##
```

Or, use a finder in your script. Each object in Oracle Sales Cloud is shipped with a set of finders. For example, the following are the available finders for the Contact REST endpoint:

- **ContactPartyNumberRF**: Finds contacts by party number.
- **MyContacts**: Finds a contact from My Contacts.
- **MyBusinessContacts**: Finds a contact from My Business Contacts.
- **MyFavoriteContacts**: Finds a contact from My Favorite Contacts.
- **PrimaryKey**: Finds a contact with the specified primary key.

The format to use a finder is:

```
?finder=<finderName>;<variableName>=<variableValue>,<variableName2>=<variableValue2>
```

Let's look at an example of using a finder. In this example, you will use the PrimaryKey finder to find a contact with the specified primary key. The variables for this finder are:

- **PartyId** (integer)
  The Oracle Sales Cloud record ID for the contact.
- **PersonProfileId** (integer)
  The unique identifier of the contact.

First, let's register the endpoint:

1. In Application Composer, under Common Setup, click **Web Services**.
2. Click the **Create Web Service Reference** icon.
3. Select **REST**, then **OK**.
4. Enter the variable name for this reference. For example, **Contact_Basic**.
5. Enter the URL for the REST endpoint that you want to integrate with. For example:

   https://host:port/crmCommonApi/resources/latest/contacts

6. For security, use basic authentication. Select **Call with basic authentication**.

7. In the Credential Key field, specify a name for the secret key that can be used to access the web service. This key name along with the user name and password is stored in the credential store.

8. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a GET method.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>GET</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Schema URL</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
</tbody>
</table>

**Code Sample**

```java
{  
  "items": [],  
  "count": 0,  
  "hasMore": false,  
  "limit": 25,  
  "offset": 0,  
  "links": [ {  
    "rel": "self",  
    "href": "http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c",  
    "name": "T1_c",  
    "kind": "collection"  
  } ]
}
```

After you register the endpoint, you can then reference the endpoint and use the PrimaryKey finder in your Groovy script. The following example illustrates the use of both a finder, the fields parameter, and the query parameter in a single call to the Contact REST endpoint. (See the next two sections for discussions about the fields parameter and the query parameter.)

```java
def conn = adf.webServices.Contact_Basic
try{  
  //Using finder and field parameters  
  def queryParams = ['finder':'PrimaryKey;PartyId=100000017340195','fields':'PartyId,PartyNumber']  
  conn.dynamicQueryParams = queryParams  
  def contacts = conn.GET()  
  println("Headers:"+conn.responseHTTPHeaders)  
  println("Status:"+conn.statusCode)  
  println("Contact after applying finder and field query parameters:"+contacts)
}
catch(Exception e){  
  println("Headers:"+conn.responseHTTPHeaders)  
  println("Status:"+conn.statusCode)  
  println("Error:"+e)
}
```
For more information about finders and their corresponding finder variables that are available for each Sales Cloud object, refer to the REST API for Oracle Sales Cloud guide.

**Passing Other Types of Parameters**

Passing a parameter or using a finder are just two ways of modifying a REST endpoint. REST APIs also support queries that can filter a collection resource through the use of the `q` and `fields` parameters.

1. **?q**
   - This query parameter defines the where clause. The resource collection will be queried using the provided expressions.
   - The format to use the ?q parameter is:
     
     \[
     \text{?q=expression1;expression2}
     \]
   - For example, maybe you want to retrieve all contacts in NY who belong to a specific department.
     
     \[
     \text{?q=Deptno>=10 \text{ and } <= 30;Loc!=NY}
     \]

2. **?fields**
   - This parameter filters the resource attributes. Only the specified attributes are returned, which means that if no attributes are specified, no attributes are returned (useful to get only the links). Use this parameter to specify the fields that you want to retrieve with this call.
   - The format to use the ?fields parameter is:
     
     \[
     \text{?fields=Attribute1,Attribute2}
     \]

For more information about additional parameters that you can attach to REST calls, refer to the REST API for Oracle Sales Cloud guide.

**Making Query Parameter Calls**

In addition to passing parameters as described earlier, you can also make query parameter calls.

You can define a query parameter and pass it directly. Or, define a payload and then pass the payload.

For example, define the query parameter itself. Note the response method is GET.

```groovy
def queryParam = [OpptyId:'6756253']
OpptyDC.dynamicQueryParams = queryParam
def response = adf.webservices.OpptyDC.GET()
```

Or, define a payload and add it to the REST endpoint. This is especially useful when trying to create or update a record. Note the response method in the following example is POST.

```groovy
def requestPayload = [OpptyName:'GreenServerTech', Account:'Pinnacle', Owner:'Lisa.Jones']
def response = adf.webservices.OpptyDC.Post(requestPayload)
```

Let's look at an example of defining a query parameter for a specific account record. In this example, you will define a query parameter and also pass the q parameter.

First, let's register the endpoint:

1. In Application Composer, under Common Setup, click **Web Services**.
2. Click the **Create Web Service Reference** icon.

3. Select **REST**, then **OK**.

4. Enter the variable name for this reference. For example, **GetAccountUsingSAML**.

5. Enter the URL for the REST endpoint that you want to integrate with. For example:

   https://host:port/crmCommonApi/resources/latest/accounts

6. For security, use the Security Assertion Markup Language (SAML) over Secure Socket Layer (SSL) authentication scheme. Select **Propagate user identity using SAML over SSL**.

7. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a GET method.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>GET</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Schema URL</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
<tr>
<td>Code Sample</td>
<td></td>
</tr>
</tbody>
</table>

```
   
   { 
   "items" : [ ], 
   "count" : 0, 
   "hasMore" : false, 
   "limit" : 25, 
   "offset" : 0, 
   "links" : [ { 
   "rel" : "self", 
   "href" : "http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c", 
   "name" : "T1_c", 
   "kind" : "collection" 
   } ] 
   }
```

After you register the endpoint, you can then reference the endpoint and, in this example, pass both the query parameter and `q` parameter in your Groovy script.

```groovy
def conn = adf.webServices.GetAccountUsingSAML
try{
    // Provide query parameter for the account object you want to receive
    def queryParams = ['q':'PartyId=300100010638186']
    conn.dynamicQueryParams = queryParams
    def accounts = conn.GET()
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Account:"+accounts)
}catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```
Creating POST Requests

You can execute standard methods such as GET, POST, PATCH, and DELETE on REST resources, using their URL. In this section, let's review POST requests.

Use a POST request to create a new item in a resource. The request content type is:

```
application/vnd.oracle.adf.resourceitem+json
```

Let's look at two POST requests. First, let's register the endpoint:

1. In Application Composer, under Common Setup, click **Web Services**.
2. Click the **Create Web Service Reference** icon.
3. Select **REST**, then OK.
4. Enter the variable name for this reference. For example, **Account_SAML**.
5. Enter the URL for the REST endpoint that you want to integrate with. For example:
   ```
   https://host:port/crmCommonApi/resources/latest/accounts
   ```
6. For security, use the SAML over SSL authentication scheme. Select **Propagate user identity using SAML over SSL**.
7. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a POST method.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>POST</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Code Sample</td>
</tr>
<tr>
<td>Code Sample</td>
<td>'{&quot;OrganizationName&quot;: &quot;M1&quot;}'</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
<tr>
<td>Code Sample</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;items&quot; : [ ],</td>
</tr>
<tr>
<td></td>
<td>&quot;count&quot; : 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;hasMore&quot; : false,</td>
</tr>
<tr>
<td></td>
<td>&quot;limit&quot; : 25,</td>
</tr>
<tr>
<td></td>
<td>&quot;offset&quot; : 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;links&quot; : [ {</td>
</tr>
<tr>
<td></td>
<td>&quot;rel&quot; : &quot;self&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;href&quot; : &quot;<a href="http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c">http://slc07pcl.dev.oraclecorp.com:9004/crmCommonApi/resources/11.1.12/T1_c</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot; : &quot;T1_c&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;kind&quot; : &quot;collection&quot;</td>
</tr>
<tr>
<td></td>
<td>} ]</td>
</tr>
</tbody>
</table>

In the next two examples, you will create a POST request to create a new contact address using the REST endpoint that you registered earlier. Both examples show you how to set the HTTP headers in your Groovy script.

1. Set the HTTP request header content type in your POST request.
```groovy
def conn = adf.webServices.Account_SAML
try{
    // Create new Account object by passing Organization name
    // Set Content-Type request header
    def OrganizationName = [OrganizationName:'TestOrganization2']
    def httpHeaders = ['Content-Type': 'application/vnd.oracle.adf.resourceitem+json']
    conn.requestHTTPHeaders = httpHeaders
    def accounts = conn.POST(OrganizationName)
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Account:"+accounts)
}
catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```

2. Set the HTTP response header content type in your POST request.

```groovy
def conn = adf.webServices.Account_SAML
try{
    // Create new Account object by passing Organization name
    // Set Content-Type request header
    def OrganizationName = [OrganizationName:'TestOrganization2']
    def httpHeaders = ['Content-Type': 'application/vnd.oracle.adf.resourceitem+json']
    conn.requestHTTPHeaders = httpHeaders
    def accounts = conn.POST(OrganizationName)
    println("Headers:"+conn.responseHTTPHeaders)
    // Retrieve Content-Type from response headers
    println("Content-Type:"+conn.responseHTTPHeaders['Content-Type'])
    println("Status:"+conn.statusCode)
    println("Account:"+accounts)
}
catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```

## Integrating with External REST Services: Explained

You can make REST outbound calls from Oracle Sales Cloud to a non-ADF REST endpoint. This topic illustrates different ways of making a call to an external REST endpoint deployed to an Oracle PaaS service, such as Oracle Java Cloud Service - SaaS Extension. In this example, the assumption is that JCS-SaaS Extension and Oracle Sales Cloud are in the same Oracle Identity Domain and are associated. This means that the user identities are in sync and trust is enabled allowing for SAML-based user identity propagation.

In this example, let’s assume that you created an external, non-ADF, trouble tickets application and deployed it to JCS-SaaS Extension. Now, you want to make calls to that resource. This topic illustrates the following:

- Retrieving trouble tickets for a given Sales Cloud account by passing an ID.
- Exception handling.
- Accessing elements in HTTP Response Headers.
Retrieving Trouble Tickets

First, let’s retrieve trouble tickets for an account from a non-ADF REST endpoint on JCS-SaaS Extension.

Let’s register the endpoint:

1. In Application Composer, under Common Setup, click **Web Services**.
2. Click the **Create Web Service Reference** icon.
3. Select **REST**, then **OK**.
4. Enter the variable name for this reference. For example, **GetTicketForAccount**.
5. Enter the URL for the REST endpoint that you want to integrate with. In this case:
   
   ```
   https://jcs-cakp.java.us2.oraclecloudapps.com/invokeTicket/troubleTicketApi/account/##AccountId##
   ```

6. For security, use the Security Assertion Markup Language (SAML) over Secure Socket Layer (SSL) authentication scheme. Select **Propagate user identity using SAML over SSL**.

7. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a GET method. (To enable the creation of new tickets in Sales Cloud, you can configure a POST method as part of this same endpoint registration.)

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>GET</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Schema URL</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
</tbody>
</table>
```

```
{  
  "accountHolder": "abc@xyz.com",  
  "requester": "Test1",  
  "assignee": "Auto Assigned User-1",  
  "share": true,  
  "subject": "New keyboard",  
  "description": "New keyboard request",  
  "status": "New",  
  "type": "Task",  
  "priority": "Urgent",  
  "tags": "New"  
}
```

After you register the endpoint, you can then reference the endpoint in your Groovy script to retrieve trouble tickets for a specific account.

The Groovy script would look like this:

```groovy
def conn = adf.webServices.GetTicketForAccount
try{
   // Provide Account Id for which user wants to retrieve trouble ticket
   def tickets = conn.GET("6637911")
   println("Headers:"+conn.responseHTTPHeaders)
   println("Status:"+conn.statusCode)
}
```

---

**Oracle Sales Cloud**
**Extending Sales**

Chapter 7
**Using Groovy Scripts**

254
Exception Handling

In this next call, let's attempt to retrieve trouble tickets from the same non-ADF REST endpoint. However, in this example, the authentication scheme will be basic authentication, and the wrong credentials will be provided. This example illustrates how the REST endpoint behaves in the case of an unauthorized request.

Let's register the endpoint slightly differently this time. In this example, use basic authentication:

1. In Application Composer, under Common Setup, click Web Services.
2. Click the Create Web Service Reference icon.
3. Select REST, then OK.
4. Enter the variable name for this reference. For example, TroubleTicketBasicAuth.
5. Enter the URL for the REST endpoint that you want to integrate with. In this case:
   ```
   https://jcs-cakp.java.us2.oraclecloudapps.com/invoke/troubleTicketApi/tickets
   ```
6. For security, use basic authentication. Select Call with basic authentication.
7. In the Credential Key field, specify a name for the secret key that can be used to access the web service. This key name along with the user name and password is stored in the credential store. For the purposes of this example, enter incorrect credentials.
8. Next, select and configure methods against the resource. You can register the resource operation (GET, POST, and so on) and the associated payload format type (JSON/XML). Only registered operations appear in the Groovy expression builder. In this example, configure a GET method.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>GET</td>
</tr>
<tr>
<td>Format</td>
<td>JSON</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Schema URL</td>
</tr>
<tr>
<td>Response Payload</td>
<td>Code Sample</td>
</tr>
</tbody>
</table>

Code Sample

```java
{
  "requester": "Mehul",
  "share": true,
  "subject": "Request for new monitor at my desk",
  "description": "Require bigger screen monitor",
  "status": "New",
  "type": "Task",
  "priority": "High",
  "tags": "Exchange"
}
```

After you register the endpoint, you can then reference the endpoint in your Groovy script to retrieve trouble tickets with the wrong credentials.
The Groovy script would look like this:

```groovy
def conn = adf.webServices.TroubleTicketBasicAuth
try{
    def tickets = conn.GET()
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Trouble Tickets:"+tickets)
}catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```

The response shows a 401 error, since your connection was created using the wrong credentials. When invocations fail, it is important to have the ability to retrieve exception headers and payloads to inspect the cause of the error. This demonstrates the ability for groovy to retrieve error payloads as well.

**Accessing Elements in HTTP Response Headers**

Finally, let's retrieve trouble tickets from the same non-ADF REST endpoint. In this final example, the authentication scheme will be basic authentication, and the correct credentials will be provided. This example illustrates how to retrieve HTTP response headers.

Let's modify the endpoint used in the previous example.

1. In Application Composer, under Common Setup, click **Web Services**.
2. Edit the **TroubleTicketBasicAuth** connection.
3. In the Credential Key field, specify the correct credentials for this call.
4. Click Save.

After you modify the endpoint, you can then reference the endpoint in your Groovy script to retrieve trouble tickets with HTTP response headers, using correct credentials.

The Groovy script would look like this:

```groovy
def conn = adf.webServices.TroubleTicketBasicAuth
try{
    def tickets = conn.GET()
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Content-Type:"+conn.responseHTTPHeaders['Content-Type'])
    println("Trouble Tickets:"+tickets)
}catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```

**Calling SOAP Web Services from Groovy Scripts**
Calling Web Services: Explained

You can call SOAP web services from your Groovy scripts in Application Composer. You might call a web service for access to internal or external data, or for example, to perform a calculation on your data.

Calling web service methods from your scripts involves two high-level steps:

1. Creating a reference to the web service. This includes registering the web service with a variable name that you use in your Groovy code.
2. Writing Groovy code in Expression Builder that calls the web service. For each call, the code must prepare the inbound arguments to the web service, call a web service method, and then process the return value from the web service.

Creating a Web Service Reference

To register a web service for use in your scripts, you first select **Web Services** in the Common Setup pane in Application Composer, then select **SOAP**. You then associate a web service variable name with a URL that provides the location of the Web Service Description Language (WSDL) resource that represents the service you want to call.

For example, you might register a web service variable name of EmployeeService for a web service that your application needs to call for working with employee data from another system. The URL for this web service’s WSDL might be:

```
http://example.com:8099/Services/EmployeeService?WSDL
```

Of course, the server name, the port number, and path name for your actual service will be different. If the port number is omitted, then it is assumed that the service is listening on the default HTTP port number 80.

Read “SOAP Web Service References for Groovy Scripts: Explained” for more information about creating web service references.

Writing Groovy Code to Call a Web Service

When you call a web service from a Groovy script, the code must prepare the arguments to the web service before calling a web service method, and then process the data returned from the web service. If your code passes structured data to and from a web service, read “Using Groovy Maps and Lists with Web Services” below.

You insert the code for the call to the web service from the **Web Services** tab in Expression Builder. As shown in the figure, the **Web Services** list displays the set of registered web service variable names and the **Functions** list displays the available methods for a given web service.

To insert a call to a web service in a Groovy script:

1. Select the **Web Services** tab in Expression Builder.
2. Select a variable name from the **Web Services** list.
3. Select a method from the **Functions** list.
4. Click the **Insert** button to insert the code to call the web service method.

As you can see in the figure, a web service call from a Groovy script has the following syntax:

```
adf.webServices.YourServiceVariableName.MethodName(args)
```

The information under function signature includes the parameter types and also the return type to indicate the type of variable the result of the call should be assigned to. The possible return types are as follows:
### Return Value vs. Return Type

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>Void</td>
</tr>
<tr>
<td>Scalar values (integer, string and so on)</td>
<td>The actual Java return type</td>
</tr>
<tr>
<td>Object</td>
<td>Map</td>
</tr>
<tr>
<td>Collection</td>
<td>List</td>
</tr>
</tbody>
</table>

### Using Groovy Maps and Lists with Web Services

When passing and receiving structured data to and from a web service, a Groovy map represents an object and its properties. For example, an Employee object with properties named Empno, Ename, Sal, and Hiredate would be represented by a map object having four key-value pairs, where the names of the properties are the keys.

You can create an empty Map object using the syntax:

```groovy
def newEmp = [:]
```

Then, you can add properties to the map using the explicit `put()` method like this:

```groovy
newEmp.put("Empno", 1234)
newEmp.put("Ename", "Sean")
newEmp.put("Sal", 9876)
newEmp.put("Hiredate", date(2013, 8, 11))
```

Alternatively, and more conveniently, you can assign and update map key-value pairs using a simpler direct assignment notation like this:

```groovy
newEmp.Empno = 1234
newEmp.Ename = "Sean"
newEmp.Sal = 9876
newEmp.Hiredate = date(2013, 8, 11)
```
Finally, you can also create a new map and assign some or all of its properties in a single operation using the constructor syntax:

```groovy
def newEmp = [Empno : 1234,
  Ename : "Sean",
  Sal : 9876,
  Hiredate : date(2013,8,11)]
```

To create a collection of objects you use the Groovy List object. You can create one object at a time and then create an empty list, and call the list’s `add()` method to add both objects to the list:

```groovy
def dependent1 = [Name : "Dave",
  BirthYear : 1996]
def dependent2 = [Name : "Jenna",
  BirthYear : 1999]
def listOfDependents = []
listOfDependents.add(dependent1)
listOfDependents.add(dependent2)
```

To save a few steps, the last three lines in the preceding example can be done in a single line by constructing a new list with the two desired elements in one line like this:

```groovy
def listOfDependents = [dependent1, dependent2]
```

You can also create the list of maps in a single operation using a combination of list constructor syntax and map constructor syntax:

```groovy
def listOfDependents = [[Name : "Dave",
  BirthYear : 1996],
  [Name : "Jenna",
  BirthYear : 1999]]
```

If the employee object in the previous codes examples has a property named Dependents that is a list of objects representing dependent children, you can assign the property using the same syntax as shown above (using a list of maps as the value assigned):

```groovy
newEmp.Dependents = [[Name : "Dave",
  BirthYear : 1996],
  [Name : "Jenna",
  BirthYear : 1999]]
```

Lastly, note that you can also construct a new employee with nested dependents all in a single statement by further nesting the constructor syntax:

```groovy
def newEmp = [Empno : 1234,
  Ename : "Sean",
  Sal : 9876,
  Hiredate : date(2013,8,11),
  Dependents : [
    [Name : "Dave",
      BirthYear : 1996],
    [Name : "Jenna",
      BirthYear : 1999]]
]
```

For more information on maps and lists, see the section called Working with Maps in Oracle Sales Cloud Groovy Scripting Reference at http://docs.oracle.com/cloud/latest/salescs_gs/CGSAC/chapter_05.htm#CGSACto_working_with_maps.
Web Service References: Explained

In the Groovy scripts that you use in Application Composer, you can include calls to both internal and external SOAP web services. For each web service that you call in your scripts, you must set up a web service reference that specifies the Web Services Description Language (WSDL) file location and the security scheme, if any, used to access the web service.

To create a web service reference, do the following in Application Composer:

1. Select Web Services in the Common Setup pane.
2. On the Web Services page, click the New icon, then select SOAP.
3. Specify a name for the web service connection.
4. Specify the URL of the WSDL file for the web service.
5. Specify the user and password credentials as required for the security scheme for the web service. Read "Specifying the Security Values for the Web Service" below for information about which schemes are supported.

Note: When registering a web service using Application Composer, the WSS security user name and password are not supported for non-SSL web services due to security issues.

After you create a web service reference, the name of the web service appears in the list available in the Web Services tab in the Expression Builder. When you select a web service from the list, you can then select any of the functions provided by the web service for use in your Groovy scripts.

You can edit existing web service references, for example, to change the security scheme used or the settings used for a particular security scheme.

Tip: When managing web service references, click the Refresh icon in the Web Services page to make sure the list is up to date. Read "Refreshing the List of Web Service References" below for information about when you need to click Refresh.

Specifying Variable Names

When you create a web service reference, you specify a variable name on the Create SOAP Web Service Connection page. This name is simply an identifier that is used in the list of web services in the Expression Builder.

Specifying WSDL URLs

The WSDL file for a web service provides information about a web service that includes the following:

- **Service.** Defines one or more ports for the web service.
- **Port.** Defines an address or connection endpoint to the web service.

For each service and port there can be one or more associated security policies.

To specify a WSDL URL:

1. On the Create SOAP Web Service Connection page, enter the WSDL file in URL format, for example: http://internal-hosted:7101/MathsWS-Model-context-root/UsernameTokenSecurity?wsdl
2. Click Read WSDL.
   - The Service, Port, and Security Scheme fields are then populated based on what is found in the WSDL. When there are multiple services and ports defined, the Service and Port fields have the first service and port found in the WSDL selected.
3. If a different service and port is required for this web service, select the appropriate values in **Service** and **Port**.

When you select a particular service and port, a default security scheme is selected based on the security policy defined in the WSDL.

If the port number is omitted, then it is assumed that the service is listening on the default HTTP port number 80.

**Specifying the Security Values for the Web Service**

For secure communication with a SOAP web service, you can use various schemes for authenticating user credentials and ensuring security. The following schemes are supported for SOAP web services from Groovy scripts:

- None
- Call with basic authentication
- Call with separate user credentials over SSL
- Call with separate user credentials and message protection
- Propagate user identity using SAML
- Propagate user identity using SAML and message protection

**Note:** If a web service is hosted on the same environment as the Groovy script that calls the web service, then the separate user name and password credentials that you provide as security values are overridden when the flow is triggered. Instead, updates are recorded as made by the signed-in user who actually called the Groovy script, not the user registered to the web service.

On the Create SOAP Web Service Connection page, you specify a credential key for the security schemes that require user name and password credentials. The web service provider will tell you about the credentials that you must use for a particular web service.

**Resolving Security Setup Errors**

You may receive some errors if some security setup has not been performed. For example, you may get a SSL certificate error when you try to create the web service reference. In this case, you must create a service request for your administrator. You must retrieve the server’s CA SSL certificate from the service provider and attach it in the service request along with the WSDL location, and error details. The administrator will import the server SSL certificate into the tenant domain and inform you when this has happened.

You may also receive errors when the web service is called from a Groovy script:

- A bad encryption error, when message protection is used
- A PolicyEnforcementException error when message protection security is used.

For these errors you must also create a service request for your administrator to resolve the errors. You must retrieve the server’s encryption certificate and the issuer certificate from the service provider and attach them both in the service request along with the WSDL location and the error details.

**Using Worked Examples of Calling Web Services from Groovy**

Worked examples of creating web service connections and calling the web service from a Groovy script are provided in separate topics as listed under "Related Links" below.

The topics cover the various security schemes that are supported for calls to both internal and external web services. The topics include information about contacting your administrator to resolve security setup errors where appropriate.
Refreshing the List of Web Service References

If new methods are added for a web service, you must click **Refresh** on the Web Services page so that the web service reference is updated. Otherwise, the new methods will not be available for the web service in the Expression Builder.

The Refresh action is applicable whenever the service contract with the client changes. This can result in new methods, changing of the signature of existing methods, and deletion of existing methods.

You might also want to click **Refresh** to display any new web service references that have been created in a separate user session.

Moving Application Changes

You can download the application changes you make in a "source" environment, and upload them into a "target" environment. This can save you time when working with changes across multiple environments.

To do this, you will do the following:

1. Use the Configuration Set Migration page to create a set of all changes and extensions made to an application environment.
2. Then, download the configuration set and upload it into another environment.
   This is often referred to as configuration set migration, or CSM.

See: "Moving Application Changes".

However, web service references created in Application Composer in the source environment won’t work in your target environment after the migration. Therefore, after you upload the configuration set to the target environment, you must recreate the web service references using Application Composer in the target environment, as well.

Web Service Calls: Explained

This topic explains how you call SOAP web services from Groovy scripts using simple examples.

You can call web services from your Groovy scripts in Application Composer, for example, to access internal or external data, or to perform a calculation on your data.

> **Note**: You can’t use Groovy scripts to create an XML/SOAP message containing attachments.

A web service call from a Groovy script has the following syntax:

```
adf.webServices.YourServiceVariableName.MethodName(args)
```

In the examples in this topic, the methods of a web service registered with the variable name EmployeeService are called.

For each web service that you call in your scripts, you must set up a web service reference in the Web Services page in Application Composer.

Retrieving an Employee by ID

The following example shows how to call a getEmployee() method of the web service by passing the integer 7839 as the single argument to the method.

```
// retrieve Employee object by id from remote system
def emp = adf.webServices.EmployeeService.getEmployee(7839)
// log a message, referencing employee fields with "dot" notation
```
println('Got employee '+emp.Ename+' with id '+emp.Empno)
// access the nested list of Dependent objects for this employee
def deps = emp.Dependents
if (deps != null) {
    println("Found "+deps.size()+" dependents")
    for (dep in deps) {
        println("Dependent:"+dep.Name)
    }
}

Creating an Employee Including New Dependents

The following example shows how to use Groovy script's convenient map and list construction notation to create a new employee with two nested dependents. The newEmp object is then passed as the argument to the createEmployee() method of the web service.

    // Create a new employee object using a Groovy map. The nested collection of dependents is a Groovy list of maps
    def newEmp = [ Ename:"Steve",
    Deptno:10,
    Job:"CLERK",
    Sal:1234,
    Dependents:[[Name:"Timmy",BirthYear:1996],
    [Name:"Sally",BirthYear:1998]]
    // Create the new employee by passing this object to a web service
    newEmp = adf.webServices.EmployeeService.createEmployee(newEmp)
    // The service returns a new employee object which may have other attributes defaulted/assigned by the service, like the Empno
    println("New employee created was assigned Empno = "+ newEmp.Empno)

Merging Updates to an Employee Object and Adding a Dependent Child Object

The following example shows how to use the mergeEmployee() method to update fields in an employee object that is retrieved at the start of the script using a call to the getEmployee() method. The script updates the Ename field on the retrieved emp object and updates the names of the existing dependents. The script then adds a dependent child object before calling the mergeEmployee() method of the web service to save the changes.

    // Merge updates and inserts on Employee and nested Dependents
    def emp = adf.webServices.EmployeeService.getEmployee(7839)
    // update employee's name to add an exclamation point!
    emp.Ename = emp.Ename + '!
    def deps = emp.Dependents
    // Update dependent names to add an exclamation point!
    for (dep in deps) {
        dep.Name = dep.Name + '!
    }
    // Add a new dependent
    def newChild = [Name:"Jane", BirthYear:1997]
    deps.add(newChild)
    emp = adf.webServices.EmployeeService.mergeEmployee(emp)

Calling an External Web Service when No Security Scheme is Required: Worked Example

This example shows how to create a connection to an external SOAP web service on the Internet and call the web service from a Groovy script used in Application Composer. The web service is not secured. For this example, the web service is used to calculate a custom field's default value.
The following table summarizes key decisions for this scenario:

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What name will you use for the web service connection?</td>
<td>mathsws</td>
</tr>
<tr>
<td>What is the URL of the Web Services Description Language (WSDL) file that you will use?</td>
<td><a href="http://external-hosted7101/MathsWS-Model-context-root/NoSecurity.wsdl">http://external-hosted7101/MathsWS-Model-context-root/NoSecurity.wsdl</a></td>
</tr>
<tr>
<td>Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.</td>
<td></td>
</tr>
<tr>
<td>Where will the web service be called from?</td>
<td>From a Groovy script expression used to calculate a custom field’s default value.</td>
</tr>
<tr>
<td>Which web service method will be called from the Groovy script?</td>
<td>getSum</td>
</tr>
<tr>
<td>This method returns the sum of two integer argument values.</td>
<td></td>
</tr>
</tbody>
</table>

To call a web service from a Groovy script when no security scheme is required, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and verify that the call succeeds.

**Prerequisites**

Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL to use from the web service provider.
2. Create a custom field for an object that has a calculated default value.
3. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

**Creating the Web Service Connection**

When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select **Web Services** in the Common Setup pane.
2. On the Web Services page, click the New icon, then click **SOAP**.
3. On the Create SOAP Web Service Connection page, enter **mathsws** in the **Name** field.
   The name must not include periods.
   After you click **Read WSDL**, the **Service** and **Port** fields are filled according to values in the WSDL file. Under **Security Scheme**, the **None** radio button becomes enabled and selected.
This figure shows the Create SOAP Web Service Connection page.

5. Click **Save and Close**.

The web service connection is created and the name and WSDL URL are listed on the Web Services page.

Adding the Web Service Call to the Groovy Script

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a **Web Services** tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the **Web Services** tab.
4. Select `mathsws` from the **Web Services** list.
5. Select `getSum` from the **Functions** list.
The code that will be inserted is shown under **Function Signature**, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click **Insert** to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click **Submit**.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

**Calling an External Web Service with Message Protection: Worked Example**

This example shows how to create a connection to an external, secured SOAP web service and call the web service from a Groovy script used in Application Composer. The web service is secured with message protection. For this example, the web service is used to calculate a custom field’s default value.

The following table summarizes key decisions for this scenario:

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What name will you use for the web service connection?</td>
<td>mathws</td>
</tr>
</tbody>
</table>

266
Decisions to Consider | In This Example
--- | ---
What is the URL of the Web Services Description Language (WSDL) file that you will use? | http://external-hosted7101/ MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecuritywsdl
This WSDL file specifies the desired message protection security scheme.

*Note:* The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.

Which credential key will you use? | mylogin

Where will the web service be called from? | From a Groovy script expression used to calculate a custom field’s default value.

Which web service method will be called from the Groovy script? | getSum
This method returns the sum of two integer argument values.

What will the server encryption alias name be? | serverenckey

Is it required to ignore the time stamp in the response from the web service? | Yes.
To ignore the time stamp, you select the **Disable Time Stamp Verification** check box. This may be required to address interoperability issues.

To call a web service from a Groovy script that is secured with message protection, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and check whether the call succeeds.
3. Contact the administrator to resolve runtime exceptions.
4. Re-create the web service connection.
5. Verify that the web service call succeeds.

Prerequisites

Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL and the user credentials to use from the web service provider.
2. Get the server encryption certificate and the Certificate Authority (issuer) certificate from the web service provider.
3. Create a custom field for an object that has a calculated default value.
4. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

Creating the Web Service Connection

When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select **Web Services** in the Common Setup pane.
2. On the Web Services page, click the New icon, then click SOAP.
3. On the Create SOAP Web Service Connection page, enter mathsws in the Name field.

The name must not include periods.

The following figure shows what happens after you click Read WSDL. The Service and Port fields are filled according to values in the WSDL file. Under Security Scheme, the Call with separate user credentials and message protection radio button becomes enabled and selected and the Credential Key and Outgoing Encryption Key fields appear.

5. Click the New Key icon next to the Credential Key field.
6. In the Create Key dialog box, enter a name in the Credential Key field, in this example, mylogin, enter the user name and password credentials supplied by the web service provider, and click OK.
7. Select Disable time stamp verification so that the time stamp in the response header from the web service is ignored.
8. Click Save and Close.

The web service connection is created and the name and WSDL URL are listed on the Web Services page.

Adding the Web Service Call to the Groovy Script

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a Web Services tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the Web Services tab.
4. Select mathsws from the Web Services list.
5. Select getSum from the Functions list.
The code that will be inserted is shown under Function Signature, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click Insert to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click Submit.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

Contacting the Administrator to Resolve Runtime Exceptions
The web service call may fail due to a number of exceptions including path certification, bad encryption, and policy enforcement exceptions. You must create a service request for your administrator to resolve the issues.

1. Create a service request for your administrator:
   a. Retrieve the server encryption certificate and the Certificate Authority (issuer) certificate from the web service provider.
   b. Attach the server encryption certificate and the issuer certificate to the service request, and include the WSDL location, and error details.
   c. Submit the service request.

The administrator will add the server encryption certificate and the issuer certificate into the Oracle Fusion CRM trust store. The administrator also creates an alias for the server encryption key, which you will use in the next task.

2. Wait until your administrator informs you that the certificates have been imported, and that the server encryption alias has been created, and then close the service request.
Re-creating the Web Service Connection

After your administrator has resolved runtime exceptions, you must re-create the web service connection and this time specify the server encryption key alias supplied by the administrator.

1. In Application Composer, select Web Services in the Common Setup pane.
2. On the Web Services page, select the web service connection you created previously, and click the Delete icon.
3. On the Web Services page, click the New icon, then click SOAP.
4. On the Create SOAP Web Service Connection page, enter mathsws in the Name field.
6. Click the New Key icon next to the Credential Key field.
7. In the Create Key dialog box, enter a name in the Credential Key field, in this example, mylogin, enter the user name and password credentials supplied by the web service provider, and click OK.
8. Select Disable time stamp verification so that the time stamp in the response header from the web service is ignored.
9. On the Create SOAP Web Service Connection page, enter serverenckey in the Outgoing Encryption Key field.
10. Click Save and Close.

The web service connection is created and the name and WSDL URL are listed on the Web Services page.

Verifying that the Web Service Call Succeeds

After you have re-created a web service connection, you must verify that the call to the web service succeeds.

1. Make sure that the Groovy script contains the code to call the web service.
2. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

Calling an External Web Service with Separate User Credentials over SSL: Worked Example

This example shows how to create a connection to an external, secured SOAP web service and call the web service from a Groovy script used in Application Composer. The web service uses a security scheme with separate user credentials and secure sockets layer (SSL). For this example, the web service is used to calculate a custom field’s default value.

The following table summarizes key decisions for this scenario:

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What name will you use for the web service connection?</td>
<td>mathsws</td>
</tr>
<tr>
<td>What is the URL of the Web Services Description Language (WSDL) file that you will use?</td>
<td><a href="https://external-hosted7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecuritywsdl">https://external-hosted7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecuritywsdl</a></td>
</tr>
<tr>
<td></td>
<td>This WSDL file specifies the desired SSL security scheme.</td>
</tr>
<tr>
<td></td>
<td>Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.</td>
</tr>
<tr>
<td>Which credential key will you use?</td>
<td>mylogin</td>
</tr>
</tbody>
</table>
Decisions to Consider | In This Example
--- | ---
Where will the web service be called from? | From a Groovy script expression used to calculate a custom field's default value.
Which web service method will be called from the Groovy script? | getSum
This method returns the sum of two integer argument values.
Is it required to ignore the time stamp in the response from the web service? | Yes.
To ignore the time stamp, you select the **Disable time stamp verification** check box. This may be required to address interoperability issues.

To call a web service from a Groovy script that is secured with SSL, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and verify that the call succeeds.

**Prerequisites**
Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL and the user credentials to use from the web service provider.
2. Get the server’s Certificate Authority (CA) SSL certificate from the web service provider.
3. Create a custom field for an object that has a calculated default value.
4. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

**Creating the Web Service Connection**
When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select **Web Services** in the Common Setup pane.
2. On the Web Services page, click the New icon, then click **SOAP**.
3. On the Create SOAP Web Service Connection page, enter `mathsws` in the **Name** field.
   
   The name must not include periods.
The following figure shows the error that is displayed after you click **Read WSDL**.

You must create a service request for your administrator to resolve the issue.

5. Create a service request for your administrator:
   - a. Retrieve the server’s Certificate Authority (CA) SSL certificate from the web service provider.
   - b. Attach the SSL certificate to the service request, and include the WSDL location, and error details.
   - c. Submit the service request.

The administrator will add the SSL certificate into the Oracle Fusion CRM trust store.

6. Wait until your administrator informs you that the SSL certificate has been imported, and close the service request.

7. Repeat steps 1 through 4.

The following figure shows what happens after you click **Read WSDL**. The **Service** and **Port** fields are filled according to values in the WSDL file. Under **Security Scheme**, the **Call with separate user credentials over SSL** radio button becomes enabled and selected and the **Credential Key** field appears.

8. Click the New Key icon next to the **Credential Key** field.

9. In the Create Key dialog box, enter a name in the **Credential Key** field, in this example, **mylogin**, enter the user name and password credentials supplied by the web service provider, and click OK.

10. Select **Disable time stamp verification** so that the time stamp in the response header from the Web service is ignored.

11. Click **Save and Close**.
The web service connection is created and the name and WSDL URL are listed on the Web Services page.

Adding the Web Service Call to the Groovy Script

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a **Web Services** tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the **Web Services** tab.
4. Select `mathsws` from the **Web Services** list.
5. Select `getSum` from the **Functions** list.

The code that will be inserted is shown under **Function Signature**, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click **Insert** to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click **Submit**.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.
Calling an Internal Web Service with Separate User Credentials over SSL: Worked Example

This example shows how to create a connection to a SOAP web service and call the web service from a Groovy script used in Application Composer. The web service uses a security scheme with separate user credentials and secure sockets layer (SSL). For this example, the web service is used to calculate a custom field’s default value.

The following table summarizes key decisions for this scenario:

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What name will you use for the web service connection?</td>
<td>mathsws</td>
</tr>
<tr>
<td>What is the URL of the Web Services Description Language (WSDL) file that you will use?</td>
<td><a href="https://internal-hosted7102/MathsWS-Model-context-root/UsernameTokenOverSSL/securitywsdl">https://internal-hosted7102/MathsWS-Model-context-root/UsernameTokenOverSSL/securitywsdl</a></td>
</tr>
<tr>
<td></td>
<td>This WSDL file specifies the desired SSL authentication scheme.</td>
</tr>
<tr>
<td>Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.</td>
<td></td>
</tr>
<tr>
<td>Which credential key will you use?</td>
<td>mylogin</td>
</tr>
<tr>
<td>Where will the web service be called from?</td>
<td>From a Groovy script expression used to calculate a custom field’s default value.</td>
</tr>
<tr>
<td>Which web service method will be called from the Groovy script?</td>
<td>getSum</td>
</tr>
<tr>
<td>Is it required to ignore the time stamp in the response from the web service?</td>
<td>Yes. To ignore the time stamp, you select the Disable time stamp verification check box. This may be required to address interoperability issues.</td>
</tr>
</tbody>
</table>

To call a web service from a Groovy script that is secured with SSL, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and verify that the call succeeds.

Prerequisites

Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL and the user credentials to use from the web service provider.
2. Create a custom field for an object that has a calculated default value.
3. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.
Creating the Web Service Connection

When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select Web Services in the Common Setup pane.
2. On the Web Services page, click the New icon, then click SOAP.
3. On the Create SOAP Web Service Connection page, enter `mathsws` in the Name field.
   The name must not include periods.
   The following figure shows what happens after you click Read WSDL. The Service and Port fields are filled according to values in the WSDL file. Under Security Scheme, the Call with separate user credentials over SSL radio button becomes enabled and selected and the Credential Key field appears.

5. Click the New Key icon next to the Credential Key field.
6. In the Create Key dialog box, enter a name in the Credential Key field, in this example, `mylogin`, enter the user name and password credentials supplied by the web service provider, and click OK.
7. Select Disable time stamp verification so that the time stamp in the response header from the Web service is ignored.
8. Click Save and Close.
   The web service connection is created and the name and WSDL URL are listed on the Web Services page.

Adding the Web Service Call to the Groovy Script

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a Web Services tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the **Web Services** tab.
4. Select `mathsws` from the **Web Services** list.
5. Select `getSum` from the **Functions** list.

The code that will be inserted is shown under **Function Signature**, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click **Insert** to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click **Submit**.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

**Calling an Internal SOAP Web Service with Message Protection**

**Security: Worked Example**

This example shows how to create a connection to a SOAP web service and call the web service from a Groovy script used in Application Composer. The web service is secured with message protection. For this example, the web service is used to calculate a custom field’s default value.

The following table summarizes key decisions for this scenario:
Decisions to Consider | In This Example
--- | ---
What name will you use for the web service connection? | mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use? | http://internal-hosted7101/ MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecuritywsdl
This WSDL file specifies the desired message protection security scheme.

Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.

Which credential key will you use? | mylogin
Where will the web service be called from? | From a Groovy script expression used to calculate a custom field’s default value.
Which web service method will be called from the Groovy script? | getSum
This method returns the sum of two integer argument values.
Is it required to ignore the time stamp in the response from the web service? | Yes.
To ignore the time stamp, you select the Disable time stamp verification check box. This may be required to address interoperability issues.

To call a web service from a Groovy script that is secured with message protection, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and verify that the call succeeds.

Prerequisites

Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL and the user credentials to use from the web service provider.
2. Create a custom field for an object that has a calculated default value.
3. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

Creating the Web Service Connection

When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select Web Services in the Common Setup pane.
2. On the Web Services page, click the New icon, then click SOAP.
3. On the Create SOAP Web Service Connection page, enter mathsws in the Name field.
   The name must not include periods.

The following figure shows what happens after you click **Read WSDL**. The **Service** and **Port** fields are filled according to values in the WSDL file. Under **Security Scheme**, the **Call with separate user credentials and message protection** radio button becomes enabled and selected and the **Credential Key** and **Outgoing Encryption Key** fields appear.

5. Click the New Key icon next to the **Credential Key** field.

6. In the Create Key dialog box, enter a name in the **Credential Key** field, in this example, `mylogin`, enter the user name and password credentials supplied by the web service provider, and click **OK**.

7. Select **Disable time stamp verification** so that the time stamp in the response header from the Web service is ignored.

8. Click **Save and Close**.

The web service connection is created and the name and WSDL URL are listed on the Web Services page.

**Adding the Web Service Call to the Groovy Script**

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a **Web Services** tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the **Web Services** tab.
4. Select `mathsws` from the **Web Services** list.
5. Select `getSum` from the **Functions** list.
The code that will be inserted is shown under **Function Signature**, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click **Insert** to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click **Submit**.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

**Calling an Internal Web Service using SAML for ID Propagation:**

**Worked Example**

This example shows how to create a connection to a SOAP web service and call the web service from a Groovy script used in Application Composer. The web service is secured by using Security Assertion Markup Language (SAML), which propagates the current user's security credentials for authentication. For this example, the web service is used to calculate a custom field's default value.

The following table summarizes key decisions for this scenario:

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What name will you use for the web service connection?</td>
<td>mathsws</td>
</tr>
</tbody>
</table>
Decisions to Consider | In This Example
--- | ---
What is the URL of the Web Services Description Language (WSDL) file that you will use? | https://internal-hosted:7102/MathsWS-Model-context-root/SamlOrUsernameTokenWithMessageProtection.wsdl

Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.

Where will the web service be called from? | From a Groovy script expression used to calculate a custom field’s default value.

Which web service method will be called from the Groovy script? | getSum
This method returns the sum of two integer argument values.

Is it required to ignore the time stamp in the response from the web service? | Yes.
To ignore the time stamp, you select the Disable time stamp verification check box. This may be required to address interoperability issues.

To call a web service from a Groovy script when SAML security is used, complete the following tasks:

1. Create the web service connection.
2. Add the web service call to the Groovy script, and verify that the call succeeds.

Prerequisites
Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL to use from the web service provider.
2. Create a custom field for an object that has a calculated default value.
3. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

Creating the Web Service Connection
When you create a web service connection, you specify a name for the web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of web services in the Expression Builder in Application Composer.

1. In Application Composer, select Web Services in the Common Setup pane.
2. On the Web Services page, click the New icon, then click SOAP.
3. On the Create SOAP Web Service Connection page, enter mathsws in the Name field.

The name must not include periods.

After you click Read WSDL, the Service and Port fields are filled according to values in the WSDL file. Under Security Scheme, the Propagate user identity using SAML radio button becomes enabled and selected.
This figure shows the Create SOAP Service Connection window.

5. Select **Disable time stamp verification** so that the time stamp in the response header from the web service is ignored.

6. Click **Save and Close**.

The web service connection is created and the name and WSDL URL are listed on the Web Services page.

### Adding the Web Service Call to the Groovy Script

In the Expression Builder dialog that you see when you create or edit Groovy scripts, there is a **Web Services** tab that lists the web services for which you have created a connection. For each web service you can include calls to the available methods in your Groovy script.

1. In Application Composer, edit the custom field that uses the expression that will contain the web service call.
2. Click the Expression Builder icon.
3. In the Expression Builder dialog, select the **Web Services** tab.
4. Select **mathsws** from the **Web Services** list.
5. Select **getSum** from the **Functions** list.
The code that will be inserted is shown under **Function Signature**, as illustrated in the figure.

6. Position the cursor at the place in the script where you want to insert the web service call.
7. Click **Insert** to insert the code to call the web service method.
8. Update the script so that two integer values are provided as arguments for the web service call.
9. Click **Submit**.
10. Verify that the web service call succeeds; in this example the custom field should have the expected default value.

**Runtime Messages: Explained**

Use the Runtime Messages page, also known as the diagnostic dashboard, to view the diagnostic messages your scripts have written to the log. Use these diagnostic messages to assist with debugging your scripts. On the Runtime Messages page, click the **Get Latest Log Messages** button to retrieve the latest Groovy script messages from the log file.

**Runtime Messages**

Runtime messages are diagnostic messages that you add to your script. They’re useful for debugging your scripts if an error occurs.

To access the Runtime Messages page in Application Composer, go to the Common Tasks pane (on the bottom-left side), then click **Run Time Messages**. You must be in an active sandbox to perform this action.
Runtime messages are user-specific. Only you can see the messages that you create.

Using a Script to Write Messages to the Log

To write messages to the diagnostic log, use the `print` or `println` function. The former writes its value without any newline character, while the latter writes its value along with a newline. For example:

```groovy
// Write a diagnostic message to the log. Notice how
// convenient string substitution expressions are
println("Status = \${Status_c}")
```

Finding Messages

To find your messages on the Runtime Messages page:

1. In Application Composer, go to the Common Tasks pane (on the bottom-left side), then click Run Time Messages
2. Click the Get Latest Log Messages button to retrieve the latest Groovy script messages from the log file.

Your `println` Groovy scripts are written to different applications depending on the UI from which they’re invoked (in other words, from where your scripts are triggered).

- If your script is triggered from the simplified UI, then the messages are written to Common Setup > Runtime Messages under the application that displays in the URL when you click the global Home icon: `http://<host>:<port>/<application>/faces/CrmFusionHome`. For example, if the application in the URL is Customer, then navigate to Common Setup > Runtime Messages under the Customer Center application to find your runtime messages.
- If your script is triggered from the desktop UI, then the messages are displayed on the same Runtime Messages page, but under the actual web application from where your script was triggered.

For example, if a `println` Groovy script is tied to a Create Quote button which is displayed on a simplified UI Opportunity page, then the message will be printed in the Customer Center application’s runtime messages. If the same action is displayed on the desktop UI Opportunity page, then the associated message will be printed in the Sales application’s runtime messages, since your script was triggered from the Sales application.

To sort in reverse chronological order so you can see your most recent messages first, click the down-pointing arrow in the Time stamp column header.

Debugging Your Groovy Scripts: Explained

Use the Groovy debugger in Application Composer to debug the object functions and validations that you defined for an object. While debugging, you can also examine object and attribute values. Access the debugger from either the Custom Objects or Standard Objects page.

Accessing the Debugger

Access the debugger from either the Custom Objects or Standard Objects page in Application Composer.

To access the debugger:

1. In Application Composer, under the Objects tree, click either the Custom Objects or Standard Objects link.
2. On the resulting Objects page for either custom or standard objects, select the object that you want to debug and then click the debugger icon in the table’s toolbar.
The debugger icon is a ladybug.

3. On the debugger UI, examine the object functions and validations defined in Groovy for that object.

Using the Debugger

The debugger contains multiple regions, described in the following table, which you can use to debug your scripts for an object:

<table>
<thead>
<tr>
<th>Debugger Region</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main toolbar</td>
<td>From the toolbar, you can select the object to examine and start the debugging process.</td>
</tr>
<tr>
<td>Left pane region</td>
<td>This region displays the object functions and validations defined for the selected object.</td>
</tr>
<tr>
<td>Main script region</td>
<td>This region displays the selected Groovy script.</td>
</tr>
<tr>
<td>Stack region</td>
<td>This region displays the call stack. For example, assume there are two functions, Function1 and Function2. Function1 calls Function2. When debugging within Function2, the Stack region displays which statement from Function2 is currently being executed, as well as information about the parent Function1 from where Function2 was called.</td>
</tr>
<tr>
<td>Variables region</td>
<td>This region displays variables and associated values.</td>
</tr>
<tr>
<td>Breakpoints tab</td>
<td>This tab displays which statement (line number) has a breakpoint. A breakpoint is a location in a Groovy script where you want the script to pause during debugging. The debugger stops at that statement.</td>
</tr>
<tr>
<td>Log tab</td>
<td>This tab displays all logs. If the script has any println() statements, then those values are captured on this tab.</td>
</tr>
</tbody>
</table>

To use the debugger:

1. In Application Composer, under the Objects tree, click either the Custom Objects or Standard Objects link.
2. On the resulting Objects page for either custom or standard objects, select the object that you want to debug and then click the debugger icon in the table’s toolbar.
3. On the debugger UI, the left pane displays the object functions and validations defined in Groovy for that object. Select the script that you want to review.

The script is displayed in the main script region.

4. To start debugging, click one of these icons in the toolbar:
   - **Step Over**
     - Review one statement in the selected script at a time.
   - **Step Into**
     - If a statement in execution is a call to some function, and you want to debug inside that function, then click **Step Into**.
   - **Step Out**
If you are debugging inside a child function and you want to move the control back to the parent function, then click **Step Out**.

- **Run**

  Move to the next breakpoint in the script. If no further breakpoints exist, then the debugger completes its evaluation of the selected script and then closes the debugger session.

### Enabling and Disabling the Debugger

The debugger is enabled by default. However, if you want to hide the debugger, or later show it again, then set the ADF: Enable Script Debugger profile option.

To set the ADF: Enable Script Debugger profile option:

1. Navigate to Setup and Maintenance.
2. From the Sales offering, search for the **Manage Administrator Profile Values** task.
3. In the Profile Display Name field, enter **ADF: Enable Script Debugger** and click **Search**.
4. In the Profile Values region, at the Site level, enter either **TRUE** or **FALSE**.

- **TRUE** displays the debugger.
- **FALSE** hides the debugger.

### FAQ for Using Groovy Scripts

#### Why did my Groovy expression time out?

In general, avoid writing Groovy scripts that might require more than 60 seconds to complete.

In Oracle Sales Cloud, a timeout of 60,000 milliseconds (60 seconds) is configured for Groovy expressions. If the expression requires more than 60 seconds to complete, an expression timeout (**oracle.jbo.ExprTimeoutException**) occurs and an error message is displayed.

For example:

```
Exception in expression "<object name>" object function <function name> : oracle.jbo.ExprTimeoutException
Expression timed out. at "<object name>" object function <function name> line <line number>
```

The location where the error message is displayed depends on where the Groovy expression was executed.

<table>
<thead>
<tr>
<th>If the Groovy expression is executed as a result of...</th>
<th>The error message appears in...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A UI operation</td>
<td>The UI.</td>
</tr>
<tr>
<td>A Web service update</td>
<td>The Web service client.</td>
</tr>
<tr>
<td>A workflow invocation</td>
<td>The error message is hidden from the end user.</td>
</tr>
</tbody>
</table>
Common situations where the Groovy expression timeout might be encountered include:

- If the Groovy expression attempts to iterate over a large collection of records, a Groovy timeout error might occur. In such a situation, any records that were initially modified are not actually committed.

  For example, let’s say that the following expression times out:

  ```groovy
def vo=newView('TestCO_c')
  vo.executeQuery()
  while(vo.hasNext()){  
    def curRow=vo.next()  
    curRow.setAttribute('F1_c',curRow.RecordName)
    println(curRow.RecordName+"":" + result)
  }
```

  Inspection of the `println` statements reveals that the `setAttribute()` operation was performed on a few records. However, if the timeout occurs while the Groovy execution is still in progress, none of the changes made are committed.

- If the Groovy expression calls a Web service, and if the Web service operation takes more than 60 seconds, a Groovy timeout error might occur. However, because the commit is part of the Web service, the commit will occur after the Web service operation is completed.

  ```groovy
def response=adf.webServices.WS.Operation(<payload>);
```
8 Creating Object Workflows

Overview

Automate business processes using object workflows in Application Composer. A business process typically has some kind of action that you want to occur in an application, based on something happening elsewhere in the application. When you create an object workflow, you indicate which business object you're impacting, such as Opportunity. You also indicate the actions that you want to happen in the application, plus the conditions that trigger these actions. Conditions are events. Once those events occur, your configured actions are immediately implemented unless you have defined a time rule (execution schedule) for one or more actions.

This chapter covers:

- The concepts and terminology used in object workflows.
- How you set trigger conditions for object workflows using groovy scripts.
- Configuring event actions such as tasks, e-mail notifications, field updates, business process flows, and outbound messages using object workflows.
- Creating an approval flow using business process composer and then triggering the approval flow through object workflows.

Object Workflows: Explained

This topic describes what object workflows are and explains the terminology related to object workflows.

You create object workflows to automate business processes. You create an object workflow with a set of actions, and set a condition to trigger these actions. This trigger is based on events. When the event occurs, all of your configured actions are run immediately unless you have set a time rule for one or more actions.
This figure illustrates how you configure and trigger object workflows.

Commonly Used Terms
Familiarize yourself with the following terms before you start working with object workflows:

- **Business Object**: A business object can be either a standard object delivered with Oracle Sales Cloud or an object that you create based on your business need.

  The business object that you use for configuring the object workflow can be either a parent object or a child object. All attributes available to you for selection when configuring a workflow belong to the object that you select for creating a workflow. This enforces that only the data relevant to the object in context is available for selection. For example, when you define a trigger condition, an expression editor lists only those fields that are relevant to the object that you have selected.

- **Active**: Indicates whether the workflow is in use. Only those object workflows which are in Active status can trigger event actions.

- **Event Point**: An event point is associated with an object and is an instance when an event occurs. Event points are of the following types:
  - When a record is created.
  - When a record is updated.

- **Event Condition**: Event point and event condition together serve as a trigger for object workflows. Event condition is an expression that supports logical, math operations, or field-value lookups. By defining an expression, you can
prevent the object workflow to trigger each time a record is created or updated. See Expression Builder in this topic for an overview of how expressions are defined.

- **Event Action:** An event action determines what action is expected from an object workflow when the conditions set for an object are met.

Event actions are of the following types:

- **Field Updates.**
  
  Use this action to specify the new values that you want to apply to the existing fields. For example, create a workflow that changes the Priority from Medium to High when the win probability exceeds 60 percent. For more information on field updates, see Object Workflows and Field Updates: How They Work Together.

- **E-Mail Notification.**
  
  Use this action to send automatic e-mail alerts. For example, create a workflow that alerts the sales team that the budget availability date has changed. For more information on e-mail notifications, see Object Workflows and E-Mail Notification: How They Work Together.

- **Task Creation.**
  
  Use this action to create and assign tasks. For example, assign a follow-up task to the owner of an opportunity when the status is still open. For more information on task creation, see Object Workflows and Task Creation: How They Work Together.

- **Outbound Message.**
  
  Use this action to send outbound message to a Web service at a specified endpoint URL. For example, send account details from one system to another. For more information on outbound message, see Object Workflows and Outbound Message: How They Work Together.

- **Business Process Flow.**

  Use this action to trigger approval flows. For example, when the Deal Amount exceeds a threshold level, the relevant sales manager receives an approval notification. For more information on business process flow, see Object Workflows and Business Processes: How They Work Together.

- **Groovy Action**

  Use this action to trigger a Groovy action from your object workflow. For example, you might trigger a Groovy action to perform cross-object updates, perform post-processing of large object hierarchies, or asynchronously initiate Web services. For more information on Groovy actions, see Groovy Scripts and Object Workflows: How They Work Together.

Here's how you access the page that you use for configuring object workflows. You must be in an active sandbox to configure an object workflow.

1. Navigate to Application Composer.
2. On the Overview page of Application Composer, click **Object Workflows**.
3. Select **Actions > Create**.

   The Create Object Workflow page opens. Use this page to start configuring your object workflow.
This figure shows the Create Object Workflow page.

**Execution Schedule**

While defining an event action, you can optionally set an execution schedule that governs when an event action should happen. Else, the event actions are run immediately. When multiple event actions are configured, the Field Updates happen first before any other event action is triggered. For event actions other than Field Updates, there is no particular sequence and the event actions are triggered based on whether or how you configure the execution schedule.

You can set a time rule for an event action based on whether that event action must occur after or before the triggering of a workflow or the occurrence of a date, and also specify the time duration in **Hours**, **Days**, or **Weeks**. If you schedule an event action for a time in the past, the event action is executed immediately after it is triggered.

This figure illustrates the execution schedule region.

**Application Composer**

To create object workflows, you use a tool called Application Composer, which is a browser-based tool that enables you to modify and extend Oracle Sales Cloud. Using Application Composer you can make the changes and see most of those changes take immediate effect in real time.

Note that all action types except E-mail Notifications are sandbox-aware, so any changes you make in Application Composer (such as creating your own fields, objects, triggers, and so on) inside a sandbox are immediately taken into consideration when an object workflow is executed. The exception is that fields you create that are used as tokens in e-mail templates must be published before they can be used.
For more information on how you modify and extend Oracle Sales Cloud using Application Composer, see Extending Oracle Sales Cloud: How It Works.

Expression Builder

Use the expression builder to write Groovy-based application logic that determines when an object workflow is triggered.

An expression builder supports building logical and math operations, including field lookups that you can optionally use to define trigger conditions. Fields in the expression builder are populated based on the object for which you are defining the workflow. The expression builder displays a warning if your expression contains an invalid attribute name; however, you must confirm whether the attribute name is actually invalid. If an attribute exists which was created at run time, then you can safely ignore the warning.

This figure illustrates the expression builder, which you can use to define Groovy conditions for object workflows.

Note: Object workflows are not triggered when records are created through file-based import. File-based import bypasses any Groovy validation and trigger logic on an object.

Some examples of the raise conditions you can use include the following:
Example 1:

\[
\text{Status=='IN\_PROGRESS' && BudgetAvailableDate==Today() + 30}
\]

Example 2:

\[
\text{if (isAttributeChanged('PrimaryContactPartyName') && PrimaryContactPartyName=='Business World') return true; else return false;}
\]

Example 3:

\[
\text{WinProb>10 || WinProb<50}
\]

Consider the following when using the expression builder to create conditions:

- Ensure that you return a valid Boolean as part of your raise condition. Returning a non-boolean value could lead to runtime errors.
- Use `return true` or `return false` to explicitly return the Boolean value and code indentation when the evaluation logic is complex, to minimize risk of runtime errors.
- For lookup values, use the lookup code instead of the display value.
- Use `<`, `>`, `==` for comparison.
- For the event point **When a record is updated**, avoid redundant calls of the actions by always specifying which field change should trigger the object workflow, using the function `isAttributeChanged()`.
- Be aware of Groovy-type coercion, if you are not returning an explicit Boolean value.
- Use the logging capability to debug your condition and review the generated log by selecting **Runtime Messages** in the Common Setup pane.

For more information on Groovy scripting, see Oracle Sales Cloud Groovy Scripting Reference at http://www.oracle.com/pls/topic/lookup?ctx=cloud132&id=CGSAC.

**Object Workflows: Examples**

Illustrated here are a few business scenarios where you employ object workflows to automate business processes.

**Scenario**

In a sales division, the management plans to set an automated business process where an opportunity must have an initial close date set automatically to 90 days from its date of creation. Being an administrator, you must create an object workflow based on management directions.
This figure illustrates the triggering of the event action for the business scenario.

To create field updates event action:

1. From the Application Composer main page, select **Object Workflows**.
2. Click **Create**.
3. Select the **Opportunity** object and provide a meaningful **Name** and **Description**.
4. Define the trigger condition using **When a record is created** event point. Use the expression builder to set the event condition as **Close Date is Null**.
5. Under Actions, select **Field Updates** event action.
This opens the Create Action: Field Updates page where you configure the event action.

6. Provide the **Name** and **Description** for the field updates action and optionally set the **Execution Schedule**.

7. Under Field Update Details, select **Close Date** and set its value to **Creation Date** plus 90 days.

8. Save the event action.

When a user creates an opportunity, the workflow is triggered which sets the close date to 90 days from the date of creation.

**Scenario**

In a sales division, the management plans to set a business process that when an opportunity is updated whose close date is current or past and the status is open, an e-mail notification is automatically sent to specified recipients and the close date is extended by 15 days. Being an administrator, you must create an object workflow based on management directions.
This figure illustrates how you configure multiple event-actions for the business scenario.

In this example, you must create a workflow with two event actions, namely, Field Updates and E-Mail Notifications.

1. From the Application Composer main page, select **Object Workflows**.
2. Click **Create**.
3. Select the **Opportunity** object and provide a meaningful **Name** and **Description**.
4. Specify the event point as **When a record is updated**. Use the expression builder to set an event condition as **Close Date** is less than or equal to **Current Date** and **Status** is **Open**.

You are now creating Field Updates event action.

1. On the Create Object Workflow page, select **Field Updates** event action.
   This opens the Create Action: Field Updates page.
2. Provide the **Name** and **Description** for the field updates action and optionally set the **Execution Schedule** for triggering the action.
3. Under the Field Update Details, select **Close Date** and set its value to **Current Date** plus 15 days.
4. Save the event action.

You are now creating the E-Mail Notification event action.

1. On the Create Object Workflow page, select **E-Mail Notification** event action.
This opens the Create Action: E-Mail Notification page.

2. Provide the **Name** and **Description** for the e-mail notification action and optionally set the **Execution Schedule** for triggering such notifications.

3. Search and select an existing **E-Mail Template** or create one with a related e-mail subject and body.

   When you create a template, you specify action-related text and field tokens that are populated at run time. See: E-Mail Templates: Explained topic.

4. Locate and specify **Specific users** under **Recipient Types**. See: recipient types in E-Mail Templates: Explained topic.

5. Click **Save**.

When a user updates an opportunity record which satisfies the trigger condition, an e-mail is sent to the specified recipients and the **Close Date** is extended by 15 days.

### Object Workflows and Field Updates: How They Work Together

This topic covers how you use object workflows to perform field updates. These updates happen automatically when the associated object workflow is triggered.

A field update is an event action that you configure when creating an object workflow. When you configure more than one event action apart from configuring field updates, field updates always run first, and then other event actions are run in no specific order.
Note: The field update invoked from Object Workflow is a background update by a Web service. If this update is performed while a record is open and you are making other updates, you get an error stating that the record has already been modified. If this happens, exit the record, reopen it, and reenter your updates. If you don’t want to do this, you can alternatively use an object level trigger or field trigger on the record.

Here’s a summary of how you configure field updates as part of object workflows:

1. From the Application Composer overview page, select Object Workflows.
2. Select Actions > Create.
3. Select an Object and provide a meaningful Name for the workflow.
4. Define the trigger condition, and select Field Updates event action.
5. On the Create Action: Field Updates page, provide a Name for the field updates action.
6. Specify the fields you want to update and the value to which you want to set those fields when the workflow is triggered. Use Update More Fields option to select additional fields to update. See Configuring Object Workflows: Example topic for an example on how you configure field updates.

Note: If you have a record open and a workflow is triggered on a field in that record, you must exit from the record and reopen it to see the updates.

Specifying Display Sequence for Fixed Choice Lists

In a fixed choice list, the list of field values can be either alphabetically arranged or have a display sequence such as High, Medium, or Low. When updating a fixed choice list field that has a display sequence, specify whether you want to populate the field with the next value, or with the previous value, in the list.

For example, for a display sequence of High, Medium, and Low, you can change from High to Medium (populate with next value) or from Low to Medium (populate with previous value) during a field update. Keep in mind, however, that if the current value is Low and you configure using populate with next value in list, the field is not updated because Low is already the last value in the sequence.

Support for Primary and Child Objects

You can configure field updates using either a parent or a child object. When you select a child object when defining an object workflow, you can update fields only in the records of that child object; you cannot update fields in the records of that child object’s parent.

For example, when defining a workflow for Opportunity Team, which is a child object of Opportunity, you can configure field updates when a new team member (child record) is added to the Opportunity Team, but you cannot configure field updates for records in the parent Opportunity object.

Additionally, when inserting field tokens while configuring field updates, only those tokens belonging to the selected child object is available for selection. See the Configuring Object Workflows: Example topic for an example of how to insert tokens when configuring field updates.

Object Workflows vs. Triggers

Object workflows are asynchronous. Use object workflows and the Field Updates action only when you need to update a field asynchronously, such as based on a time delay.

If you want changes to be immediately reflected on the user interface at run time, however, use scripting and triggers, such as the Before Update trigger.
Object Workflows and E-Mail Notifications

Object Workflows and Email Notification: How They Work Together

This topic covers how you create email notifications action when creating object workflows. Email notifications send out automated alerts to the specified recipients when the associated object workflow is triggered. You can configure email notifications using either a parent or a child object.

For example, you can define a workflow that when the win probability of an opportunity is set to 30 percent or more, the workflow automatically sends out email alerts to the resources who are assigned to that opportunity.

Here’s a summary of how you typically configure an email notification action as part of object workflows:

1. From the Application Composer overview page, select Object Workflows.
2. Select Actions > Create.
3. Select an Object and provide a meaningful Name for the object workflow.
4. Define the trigger condition, and select E-Mail Notification as the event action.
5. On the Create Action: Email Notification page, provide a Name for the email notification action.
6. Search and select an existing email template or create one.

When you create a template, you specify action-related text and field tokens that are populated at run time. See Email Templates: Explained.

7. Specify the Recipient Types. See: Recipient Types in this topic.

See Configuring Object Workflows: Example topic for an example of how you configure email notifications and email templates.

Email Templates

Email notifications are based on email templates, which define the layout of the emails and ensure that the notifications triggered by the same type of business event have consistent look and feel. See Email Templates: Explained topic for more information on creating email templates.

Recipient Types

The recipients available for sending the email notifications are contextual. If you are defining a workflow using a parent object, the recipients belong to the parent and its related child objects. If you are defining a workflow using a child object, then the recipients belong only to that child object. Recipient types include:

- From address
- To address
- Reply-to address
- Cc address
- Bcc address

You use field tokens to insert runtime values of fields. When inserting such tokens, only those fields belonging to the selected primary or child object are available for selection. See "Configuring Object Workflows: Example" for an example of how you select and insert field tokens.
The following are the recipient types:

- **Fields on record**: If the selected object contains user-defined text fields that store emails, you can select those fields to send email notifications (for example, Primary Contact Email under Opportunity object). The user-defined text field can store a single email address or multiple comma-separated email addresses.

- **Relative users on record**: Contains relative users who could be either a creator of an opportunity record, a person who last updated a record, a manager or his directs, or a team of resources working on a project, and so on.

- **Roles**: Contains users assigned to a particular resource role. The email notification is sent to all enterprise groups or users that have been granted that resource role.

  When selecting a Role, you can optionally apply an additional filter to specify the organization to which that role belongs. For example, if you want to send an email notification to a Sales Director role within a particular organization, specify that organization using the **Filter By** field in the dialog where you select the role.

- **Resources**: Resources associated with the record, such as Created By or Last Updated By.

- **Resource Groups**: Groups of resources associated with the records, such as Direct reports of Created By, or Subordinates of Last Updated By.

- **Specific users**: Contains details of individual users and their emails.

- **Specific email addresses**: Enter individual emails separated by a comma.

> **Note**: The Search function for specific users includes a **Search** button to initiate the search; for the other recipient types, such as Roles, the application performs an automatic search after you type in your search terms and click or tab out of the search box.

### Specifying Complex Notification by Email

To specify a complex email notification (where the email address is calculated rather than merely selected from a list or dynamically generated), you must define a field of type Formula to calculate the email address.

You can then use the Fields on record recipient type to select the user-defined field containing the formula.

For example, each time an opportunity is updated, you might want to send an email to notify a user saved in a user-defined object in Opportunity. You would do this by creating your own Formula text field on the Opportunity object to specify the user, then select this field using the Fields on record recipient type in the object workflow.

### E-Mail Templates: Explained

This topic describes how you create, manage, and use e-mail templates. You use these e-mail templates for configuring the e-mail notifications action in object workflows. E-mail templates define the layout of the e-mails, which ensures that e-mail notifications triggered by the same type of business event for a specific object have a consistent look and feel.

You cannot create e-mail templates while inside an active sandbox.

### Configuring E-Mail Templates

You create a template for use with a particular type of object. You then use this template only for that object when configuring e-mail notifications. You can either create e-mail templates or create templates using any e-mail editor, and then upload them for use. You can create e-mail templates, for example, for including your company letterhead in outbound e-mail communications.
You can create e-mail templates in Application Composer in two ways:

1. When configuring e-mail notification action.

   On the Create Action: E-Mail Notification page, select the Create E-Mail Template icon in the E-Mail Template field.

2. Using the E-Mail Templates link on the Overview page of the Application Composer.

This figure shows the Create E-Mail Template page.

For an example on how you create e-mail templates, see Configuring Object Workflows: Example topic.

Here’s what you can or can’t do when creating e-mail templates:

- You can perform basic formatting such as font format, character format, paragraph alignment, bullets and numbering, and so on as in any HTML based e-mail client.

  Templates are automatically converted to plain text for users who cannot view HTML e-mails.

  For advanced HTML editing, use the Source Code Editing Mode, which is available as an icon under the E-Mail Body region. Under this mode, the tags are set to visible. You can copy and edit the source code in any HTML editor, and then paste the edited code back. Similarly, you can use the Rich Text Editing Mode, which is available next to the Source Code Editing Mode icon.

- You can attach artifacts relevant to the e-mail template. Attachments are included in every e-mail, which uses the template that has an attachment. You can also send attachments as links to avoid crowding the inbox of the recipients. An attachment can be a file on a local computer or a shared file in a repository.

  **Note:** File attachments cannot exceed 10 MB.
• Specify whether an e-mail template is Active. You can use only Active templates to create e-mail notifications.
• You can insert fields or functions in the e-mail body. You use functions to insert date, current date and time, or a hyperlink to a record. You can also Browse and then insert the content of a local file in the body of the e-mail template.
• You can link directly to object details pages using primary key fields of the primary object records or using foreign keys for related objects. You can select these primary key and foreign key fields in the field drop-down list to substitute them into the direct links as tokens. The following objects are supported: Opportunity, Lead, Account, Contact, Household, Activity, and any objects you create. Refer to the description of direct linking for details on how to construct these links.
• You can specify the tokens or fields, which are populated with the values at run time.

**Note:** You can’t use fields of type Dynamic Choice List as tokens.

### Managing E-Mail Templates
Managing e-mail templates include tasks that enable you to search, view, duplicate, create, edit, or delete a template.

To manage e-mail templates, select the E-Mail Templates link on the Overview page of the Application Composer.

This figure shows the E-Mail Templates page, which you use for managing e-mail templates.

Options available for managing e-mail templates include:

• Viewing the existing e-mail templates and modifying views.
• Filtering or querying existing templates including partial search using wildcard.
• Editing an existing template or creating a duplicate. You cannot change the object for which a template is defined.
• Creating a template.
• Detaching the listed templates for a full-page view.
• Viewing which templates are Active.

**Note:** You can use only Active templates for configuring e-mail notifications. Saving does not automatically make the template Active.
Related Topics

- Direct Page Links: Explained

Object Workflows and Tasks

Object Workflows and Task Creation: How They Work Together

You can configure object workflows to trigger auto-creation and assignment of tasks.

You can define tasks action for only those objects that support the creation of tasks. For other objects, the tasks option is disabled.

Here’s how you can create a task:

1. Navigate to Application Composer.
2. From the Overview page, click Object Workflows.
3. Click Create.
4. Select an Object and provide a meaningful Name and Description.
5. Define the trigger condition using When a record is created or When a record is updated event point.
6. Under the Actions region, click Create for Tasks Creation.

This opens the Create Action: Tasks Creation page.

7. Provide the Name and Description for the event action and optionally set the Execution Schedule.
8. Use the Task Details region to configure a task based on your requirements. Here’s how you can use the tasks details region:

- Select a Subject and Description, as appropriate, for your task notification.
- Select the Due Date and Start Date for a task. When you select a date, you can also provide logical conditions for these dates, for example, a Due Date must be 30 days after the Start Date.
- Select an Owner for the task. The values in this list change depending on the object for which the task is being defined. A task can have only one owner: either an individual person or a single resource. You can search for users by name or user ID. You can store a user’s PartyID or GUID in a text field you create and use the value as the task owner.
- Select the Resources for the task. The values in these lists change depending on the object for which the task is being defined. Resources can be owner, resource team, resource team with different access levels, member functions, and so on. You can store multiple comma-separated PartyIDs or GUIDs in a text field you create and use the values as task resources.
- Select a Primary Contact for the task. You can select one from contacts related to the object record, or specify one by name. If the selected object contains user-defined text fields that store the contact’s name or contactID, you can select those fields to be set as primary contact.
- Select the Type to which a task belongs. This field uses FND lookup for values. The delivered values are call, chat, demo, e-mail, and meeting.
- Assign a Priority to the task. The default priority is Medium.
9. If a standard object or an object you created has more than one one-to-many relationships defined through dynamic choice list or reference relationship with the Activity you have chosen, the Related To field appears at the bottom. This Related To field shows the first created relationship as default.

   You can select this field only once for the object in context, after which the field appears as read only.

10. Save the event action.

   **Note:** If an object for which you are defining this event action contains a customer, that customer is automatically included in the created task.

For an example of how you configure tasks, see Configuring Object Workflows: Worked Example topic.

### Creating Tasks Using Child Objects

When you configure an object workflow using a child object, the fields that you select are specific only to that child object.

When a child-object based task is triggered, the task is assigned to the parent object of that child. You cannot configure a task for a child object of another child object.

### Creating Tasks Using Objects with No Associated Activity

You can create a task for a standard object or an object you created even if it doesn't have a relationship with an Activity. If you try to create a task on an object with no relationship to an Activity, you see a message letting you know that a standalone task will be created unless you first create such a relationship. After they are created, standalone activities will show in the task owner's Activity work area in the desktop UI, and in the top-level Activities card in the simplified UI.

### Related Topics

- Object Relationships: Explained

### Enabling Task Creation for Object Workflows: Worked Example

You can configure object workflows to trigger the automatic creation and assignment of tasks. For example, you might want to remind a sales representative to call a customer one month before their contract ends. Most objects support the creation of tasks. However, for any object (for example, Asset) that doesn't have a standard relationship to Activity as delivered, the Task event action is disabled. To enable the Task event action, you create a relationship between the target object and Activity objects.

Create a relationship between the desired object and the Activity object in one of two ways:

- Create your own dynamic choice list field for the Activity object that points to the desired object's records.
- Create a one-to-many reference relationship with desired object as the source, and the Activity object as the target.

   **Tip:** This type of relationship is similar to a dynamic choice list relationship. The difference is that when you want to create your own relationship between the objects but you don't need to show it in the UI as a choice list, then you might prefer to simply create the one-to-many relationship rather than the dynamic choice list relationship. With the one-to-many relationship, you don't get a choice list field for the object to add to any Activity user interface.

Once a dynamic choice list or reference relationship exists, you can create tasks for object workflows associated with the desired object.
Creating a Dynamic Choice List Field

In this example, let’s create a dynamic choice list field for the Activity object that is populated by records from the Asset object.

1. In Application Composer, select the Activity object.
2. Select Fields.
3. Create a field of type Choice List (Dynamic).
4. Complete the general steps to configure a new dynamic choice list field. For example, set the display label for the field.
5. In the Related Object list, select the Asset object.
6. Complete the rest of the steps to create this dynamic choice list field. For example, in the List Selection Display Value field, select the field that displays as the first column in the dynamic choice list at run time.
7. Click Submit.

Creating a Relationship

In this example, let’s create a relationship between the Activity and Asset objects.

1. Navigate to Application Composer.
2. Click the Relationships link under Common Setup.
3. Click the Create icon.
4. In the Source Object field, select the Asset object.
5. In the Target Object field, select Activity.
6. In the Cardinality field, select 1:M.
7. Click Save and Close.

Related Topics

- Object Relationships: Explained
- Dynamic Choice Lists: Explained

Object Workflows and Outbound Messages

Object Workflows and Outbound Message: How They Work Together

You can configure an object workflow to send an outbound message to a Web service at a specified endpoint URL. The endpoint URL is an external Web service that shares data with Oracle Sales Cloud, and must conform to the service WSDL of Oracle Sales Cloud.

For example, you can define an object workflow that automatically sends, based on a trigger, an outbound message containing an object like opportunity, lead, or account details from one system to another.
This figure illustrates how outbound messages are configured as part of object workflows.

You can configure outbound-message event for a parent or a child object. When you define a workflow using a parent object, the outbound message is sent using service data objects (SDO) of that parent object as well as its child objects. When you define a workflow using a child object, the outbound message is sent using the SDO of only the child object.

To define an outbound message action:

1. From the Application Composer main page, select **Object Workflows**.
2. Click **Create**.
3. Select an **Object** and provide a meaningful **Name** and **Description** for the workflow you are creating.
4. Define the trigger condition using **When a record is created** or **When a record is updated** event point.
5. Select the **Outbound Message** event action.

   This opens the Create Action: Outbound Message page.

6. Specify the **Name** and **Description** for the outbound message action and set the **Execution Schedule** for triggering the action. See: Execution Schedule in this topic.
7. Provide the **endpoint URL** of the external Web service.

   To build the external Web service, use the OutboundMessageService.wsdl and .xsd files. Then, when defining the outbound message event action for the object workflow, you provide the endpoint URL of the external Web service.

   For example, an endpoint URL can be `http://GlobalFusion:7011/OMTestOpportunity/OutboundMessageServiceSoapHttpPort`. At run time, a service data object containing details of the object on which the object workflow is defined is sent to the specified endpoint URL.

8. Save the event action.
Execution Schedule

While defining an event action, you can optionally set an execution schedule that governs when an event action should happen. Else, the event actions are run immediately. When multiple event actions are configured, the Field Updates happen first before any other event action is triggered. For event actions other than Field Updates, there is no particular sequence and the event actions are triggered based on whether or how you configure the execution schedule.

If a field update event action is also scheduled along with outbound messages event action, the field updates event action is triggered first, so that the outbound messages contain the updated data.

Security Considerations

The outbound messages can use either an authentication-only client-side security policy or a transport-level security policy that protects the message during transfer. The default authentication-only policy used by object workflow outbound message is `oracle/wss10_saml_token_client_policy`. This policy includes Security Assertion Markup Language (SAML) tokens in outbound Simple Object Access Protocol (SOAP) request messages. This policy should only be used when the target web service is located within a secure network segment.

The corresponding service can use any compatible service policy, such as `oracle/wss10_saml_token_service_policy` or `oracle/wss_saml_or_username_token_service`.

When the service is outside your firewall, you must protect the message by selecting the Protect Message option on the Create Action: Outbound Message page. When protected, the outbound message uses `oracle/wss_username_token_over_ssl_client_policy` with one-way secure socket layer (SSL) enabled by default. The corresponding service must use a compatible service policy, such as `oracle/wss_username_token_over_ssl_service_policy`.

Outbound Messages Protection

You must perform the following setup activities when you are using the Protect Message option:

1. Oracle Sales Cloud Applications requires credentials to access the remote system to which the outbound message is being sent. Log a service request (SR) to provide these credentials to Oracle Sales Cloud Applications to access the remote system.
2. If the remote system is using a self-signed SSL certificate, log an SR for the Oracle administrator to import the SSL certificate into the Oracle Sales Cloud Applications.

Creating Object-Specific Web Services: Explained

You can configure an object workflow to send an outbound message to a third-party Web service at a specified endpoint URL. An endpoint URL is an external Web service that receives data from Oracle Sales Cloud applications. The third-party Web service must conform to the service WSDL defined by Oracle Sales Cloud.

To build the external Web service, use the OutboundMessageService.wsdl and .xsd files. Then, when defining the outbound message event action for the object workflow, you provide the endpoint URL of the external Web service.

This topic provides the following:

- How to configure a Web service
- An example of the OutboundMessageService.wsdl
- An example of the OutboundMessageService.xsd
- The security policies available
Configuring a Web Service

To configure a Web service, you must replace the parameters in the OutboundMessageService.xsd file using the Oracle Sales Cloud Web service instructions.

For more information about Web services, see the SOAP Web Services for Oracle Sales Cloud guide.

1. For a standard object, search the guide by object name.
   For custom objects, search for the generic Web service for all custom objects in the corresponding application, for example, Sales Custom Business Object.
2. Extract the .xsd files from the live environment URL using the information provided under the service.
3. Replace the parameters in OutboundMessageService.xsd with the names for the object of your interest.
   The parameters are marked in the .xsd file as parameters, $OBJECT_TARGET_NAMESPACES$ and $OBJECT_NAMES$.

Examples of OutboundMessageService.wsdl and OutboundMessageService.xsd are provided here for reference.

WSDL File Example

This section contains an example of the OutboundMessageService.wsdl, for your reference.

```xml
//Sample OutboundMessageService.wsdl
<wsdl:definitions
    name="OutboundMessageService"
    targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/"
    xmlns:errors="http://xmlns.oracle.com/adf/svc/errors/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns:types="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
>
    <wsdl:types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
                schemaLocation="OutboundMessageService.xsd"/>
        </schema>
    </wsdl:types>
    <wsdl:message name="OutboundMessageService_processOutboundMessage">
        <wsdl:part name="parameters" element="types:processOutboundMessage"/>
    </wsdl:message>
    <wsdl:message name="OutboundMessageService_processOutboundMessageResponse">
        <wsdl:part name="parameters" element="types:processOutboundMessageResponse"/>
    </wsdl:message>
    <wsdl:portType name="OutboundMessageService">
        <wsdl:operation name="processOutboundMessage">
            <wsdl:input message="tns:OutboundMessageService_processOutboundMessage"/>
            <wsdl:output message="tns:OutboundMessageService_processOutboundMessageResponse"/>
            <wsdl:fault name="ServiceException" message="errors:ServiceException"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="OutboundMessageServiceSoapHttp" type="tns:OutboundMessageService">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="processOutboundMessage">
            <soap:operation soapAction="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/processOutboundMessage"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```
XSD File Example

This section contains an example of the OutboundMessageService.xsd, for your reference. The parameters are marked in the .xsd file as parameters, $OBJECT_TARGET_NAMESPACE$ and $OBJECT_NAME$.

//Sample: OutboundMessageService.xsd

```xml
<schema elementFormDefault="qualified" targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/">
  <import namespace="http://xmlns.oracle.com/adf/svc/errors/" schemaLocation="ServiceException.xsd"/>
  <import namespace="http://xmlns.oracle.com/adf/svc/types/" schemaLocation="BC4JService.xsd"/>
  <import namespace="$OBJECT_TARGET_NAMESPACE$" schemaLocation="$OBJECT_NAME$.xsd"/>
  <element name="processOutboundMessage">
    <complexType>
      <sequence>
        <element name="object" type="ns1:$OBJECT_NAME$"/>
      </sequence>
    </complexType>
  </element>
  <element name="processOutboundMessageResponse">
    <complexType>
      <sequence/>
    </complexType>
  </element>
</schema>
```

Security Policy

The outbound messages can use either an authentication-only client-side security policy or a transport-level security policy that protects the message during transfer. The default authentication-only policy used by object workflow outbound message is oracle/wss10_saml_token_client_policy. This policy includes Security Assertion Markup Language (SAML) tokens in outbound Simple Object Access Protocol (SOAP) request messages. This policy should only be used when the target web service is located within a secure network segment.

The corresponding service can use any compatible service policy, such as oracle/wss10_saml_token_service_policy or oracle/wss_saml_or_username_token_service.

When the service is outside your firewall, you must protect the message by selecting the Protect Message option on the Create Action: Outbound Message page. When protected, the outbound message uses oracle/wss_username_token_over_ssl_client_policy with one-way secure socket layer (SSL) enabled by default. The corresponding service must use a compatible service policy, such as oracle/wss_username_token_over_ssl_service_policy.
Object Workflows and Business Processes

Object Workflows and Business Processes: How They Work Together

This topic explains how you use object workflows to trigger business process flows. When you configure object workflows, you also specify the actions that the workflow must perform when triggered. One of the actions you configure is the Business Process Flow action.

Before you begin, you might want to read Object Workflows: Explained and its related topics to familiarize yourself with what object workflows are, how you set trigger conditions, and the event actions that the object workflows support.

Overview

A business process flow in object workflows is essentially an approval flow. You use these approval flows to send approval requests, for example, when the following events occur:

- When the win probability of an opportunity is changed above a threshold level.
- When an object you created has been updated.
- When a new order is created.

You configure a business process flow action in object workflows using Application Composer; however, the underlying process of creating an object workflow, creating an approval flow, associating the object workflow with the approval flow, triggering them both to send approval requests, and then taking an action on the requests involves multiple applications.
This figure illustrates how object workflows and business processes work together, and the various applications involved in the end-to-end flow.

- Application Composer. You use this application to configure object workflows.
  For examples of how you configure various event actions in object workflows, see Configuring Object Workflows: Worked Example and Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example topics.
- Oracle Business Process Composer. You use this application to create business processes (or approval flows) using existing or user-defined templates.
For an example of how you create and deploy a business process flow, see Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example topic.


- Oracle Business Process Management (BPM) Worklist application. You use this application to take appropriate action on the approval requests.

The Worklist application displays tasks or approvals that are assigned to a user. Your worklist tasks appear on the Home page of Oracle Sales Cloud.

Configuring a Business Process Flow Action in Object Workflows

Object workflows contain a set of actions that are run when the workflow is triggered. One of these actions is the Business Process Flow action.

To configure business process flow action:

1. Navigate to Application Composer.
2. On the Overview page, click Object Workflows.
3. Select Actions > Create.

Create Object Workflow page opens.

4. Specify a Name for your workflow.
5. Specify the trigger condition in the Event Point and Condition region.

Create Action: Business Process Flow page opens. You use this page to configure the business process flow action.
This figure shows the Create Action: Business Process Flow page:

Create Action: Business Process Flow

- **Object**: Opportunity
- **Type**: Business Process Flow
- **Name**: Win Probability is more than forty percent

### Execution Schedule

#### Business Process Flow Details

<table>
<thead>
<tr>
<th>Project Name</th>
<th>ExtnBusinessProcessComposite</th>
</tr>
</thead>
</table>

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>ihussain</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>'approvalStatus_c'</td>
</tr>
<tr>
<td>emailAddress</td>
<td><a href="mailto:cola@pinnacle.com">cola@pinnacle.com</a></td>
</tr>
<tr>
<td>title1</td>
<td>Name</td>
</tr>
<tr>
<td>title2</td>
<td>Region</td>
</tr>
</tbody>
</table>

The key configurations that you perform using the Create Action: Business Process Flow page are as follows:

- Provide a meaningful **Name** for your business process flow action.
- Click the Search icon in the **Project Name** field to open Search and Select: Business Process Flow dialog.
This figure shows the search and select dialog.

You use the search and select dialog to do the following:

- Search and specify the project (or approval flow) to trigger when the object workflow is triggered.
- Launch the Oracle Business Process Composer to create a project based on the chosen template.

To launch the composer, you click the Create icon to open Create Business Process Flow dialog. You then specify the **Name** for your project, and then click **OK**.

- Specify the input parameters that you want to supply to the approval process. See "Creating an Approval Flow Using Business Process Composer" section in this topic.

Creating an Approval Flow Using Business Process Composer

You access Oracle Business Process Composer using either of the following ways:

- Using the Business Processes link on the Overview page of the Application Composer.

Oracle provides a set of business processes you can use without modification, or you can use these business processes as a template to create your own version of the processes. You typically create approval flows using ExtnBusinessProcessComposite template, which is the standard or default template delivered with the product. If there are other templates available, create your approval flow based on the relevant template.

**Note:** The ExtnBusinessProcessComposite template supports only Business Process Management Notation (BPMN), which is an industry standard notation for defining business processes. For more information on BPMN, see [http://www.bpmn.org](http://www.bpmn.org).

You can only use the existing templates (either as provided or by copying and editing them) to create projects using the services, tasks, and business rules as provided in the template. For more information on the considerations involved when using templates, see Object Workflows and Business Processes: Critical Choices.
For an example of how you create an approval flow using ExtnBusinessProcessComposite template, see Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example.

**Associating the Object Workflow with the Relevant Approval Flow**

While configuring the business process flow action, you use the input parameters to specify the field-values to pass from the object workflow to the chosen approval flow when the object workflow is triggered.

The input parameters that you pass from object workflows to the approval flow depends on the template that you have chosen for creating the approval flow. When you use the default ExtnBusinessProcessComposite template to create an approval flow, here’s how you typically configure the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>Specify the people who should receive the approval request. You can specify one or more of the following:</td>
</tr>
<tr>
<td></td>
<td>- Resources associated with the record (such as its owner or creator, the manager of the owner, or the last person to update the record).</td>
</tr>
<tr>
<td></td>
<td>- Resource groups (such as the direct reports or subordinates of the owner).</td>
</tr>
<tr>
<td></td>
<td>- Specific users.</td>
</tr>
<tr>
<td></td>
<td>- Fields on record: If the selected object contains user-defined text fields that store the approver’s GUID, you can select those fields to send approval notifications to a single user or comma-separated GUIDs of multiple users.</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>Specify the field that displays the status of the approval request.</td>
</tr>
<tr>
<td>emailAddress</td>
<td>Optionally specify the e-mail address of one or more individuals who must receive e-mails on the outcome of the approver action.</td>
</tr>
<tr>
<td>title1</td>
<td>Optionally append more information to the title of the notification.</td>
</tr>
<tr>
<td>title2</td>
<td>Optionally append more information to the title of the notification.</td>
</tr>
</tbody>
</table>

For more information on the considerations involved when using these input parameters, see Object Workflows and Business Processes: Critical Choices topic.

**Triggering the Object Workflow and the Associated Approval Flow**

When you trigger the object workflow which has an associated approval flow, the approval flow is also triggered. For example, when you trigger the approval flow that you created using ExtnBusinessProcessComposite template, a task appears in the BPM Worklist application for the person who you have specified in the approver field under input parameters.

You can also configure these input parameters to modify some of the notifications, but with some limitations on what you can modify and how.

For more information on the types of notifications available and how you can modify them, see Object Workflows and Business Processes: Critical Choices.
Object Workflows and Business Processes: Critical Choices

This topic covers the key points to consider when you work with object workflows to trigger a business process. When you configure object workflows, you also specify the actions that the workflow must perform when triggered. One of the actions you configure is the Business Process Flow action. This business process flow action in object workflows is essentially an approval flow.

Before you begin, you might want to read Object Workflows: Explained topic to familiarize yourself with what object workflows are, how you set trigger conditions, and the event actions that the object workflows support. Additionally, to know how object workflows and business processes interact with each other, read Object Workflows and Business Processes: How they Work Together topic.

Overview

You use templates in Oracle Business Process Composer to create and deploy a project, which you then call from object workflows. A project that you create and deploy is called an approval flow. You must create and deploy at least one project (or approval flow) before you configure the Business Process Flow action in object workflows.

Note: You must work directly in the mainline metadata to create an approval flow. You can't configure the business process flow task while you're in an active sandbox.

You configure a business process flow action in object workflows using Application Composer; however, the underlying process of creating an object workflow, creating an approval flow, associating the object workflow with the approval flow, triggering them both to send approval requests, and then taking an action on the requests involves multiple applications.

- Application Composer. You use this application to configure object workflows.
  For examples of how you configure various event actions in object workflows, see Configuring Object Workflows: Worked Example and Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example topics.
- Oracle Business Process Composer. You use this application to create business processes (or approval flows) using existing or custom templates.
  For an example of how you create and deploy a business process flow, see Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example topic.
- Oracle Business Process Management (BPM) Worklist application. You use this application to take appropriate action on the approval requests.
  The Worklist application displays tasks or approvals that are assigned to a user. Your worklist tasks appear on the Home page of Oracle Sales Cloud.

Using the Default Template to Create Approval Process

The template delivered with the product is the ExtnBusinessProcessComposite template, which uses a basic CrmCommonSerialApprovalProcess process flow. This template contains an approval process covering a typical Oracle Sales Cloud use case. Any project that you create with the seeded template uses CrmCommonSerialApprovalProcess process.
Note: The ExtnBusinessProcessComposite template supports only Business Process Management Notation (BPMN), which is an industry standard notation for defining business processes. For more information on BPMN, see http://www.bpmn.org.

This figure shows the standard ExtnBusinessProcessComposite template:

For an example of how you create and deploy a project, see Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example.

The default ExtnBusinessProcessComposite template contains services, business rules, and tasks which you can use for configuring approval flows. You can change or add nodes or business rules for human task implementation process.

For example, you can change SerialGivenUserNameApproval with SerialApprovalGroupApproval. However, when creating or modifying a project, do not attempt the following unless you are sure:

- Deleting or modifying the services, rules, tasks, or system fields in an existing business process. It may fail validations.
- Removing or changing reserved parameters such as owner, heldEntity, heldEntityId, heldEntityStatusField, and objectProperties.
- Removing nodes such as CreateHold, UpdateHold, UpdateEntity, and so on.

Note: You can edit only those projects, which have been created by launching BPM Composer from within the Application Composer.

To see the services, tasks, and rules available, select the Project Home tab in Oracle Business Process Composer.

This figure shows the services, tasks, and rules available in the template.
Supping Inputs to the Business Process from Object Workflows

Using input parameters you can pass specific values to the associated approval process when the trigger condition is met. These input parameters are specific to the template that you are using to create your approval flow. By default, you use the ExtBusinessProcessComposite template, which is shipped with the product and covered in the documentation related to object workflows. If you are using some other template, refer to the documentation for that product for the parameters that you can use.

To see where these input parameters reside in Oracle Business Process Composer, right click on the Start node of the approval flow and select Properties.
This figure illustrates where you can see the input parameters in business process composer.

Notice the heldEntityStatusField and approvers parameters in the figure. These are some of the parameters you are mapping to from object workflows when using the ExtnBusinessProcessComposite template.

When using the standard ExtnBusinessProcessComposite template, these are the inputs you can supply to the associated approval process:

- approvers
- heldEntityStatusField
- emailAddress
- title1
- title2

While the approvers and heldEntityStatusField parameters are mandatory, the emailAddress, title1, and title2 are optional parameters.

For more information on these inputs and how you can configure notifications using these inputs, see "Configuring Input Parameters for Modifying Notifications" section in this topic.
Using input parameters, you can pass either static or runtime values to the approval process as follows:

- To pass actual or static values to the approval flow, select either **Literal** check box in the Select Default Value dialog or manually enter parameters in single quotes, for example, ‘user1’.
  
  This figure illustrates how you select and mark the input parameters as Literal. This dialog appears when you click on the function icon in the Inputs region.

- To pass runtime values, do not select Literal in the Select Default Value dialog or just enter the parameters manually without quotes.
  
  For example, to pass a runtime field (token) for Opportunity Region, enter Region (without quotes). When the workflow is triggered, this field is replaced with runtime value of Region.

You can also obtain approvals serially or in parallel from multiple approvers. For example, to obtain serial approvals first from user1 and then from user2 using Literal values, enter ‘user1,user2’. Similarly, for runtime fields, enter user1,user2.

**Note:** To supply multiple, literal values use single quotes for the entire string without any space in between those values.

### Configuring Input Parameters for Modifying Notifications

The field values or the input parameters that you pass from object workflows to the approval flow depends on the template that you have chosen for creating the approval flow. This section contains examples assuming you are using the default ExtnBusinessProcessComposite template.

Before we go into configuring input parameters, it is essential to understand what types of notifications occur when the approval flow is triggered, and when the approval flow is approved or rejected.

- When the approval flow is triggered, the BPM worklist notification shows, for example, “Opportunity has been submitted for your approval”, which is in format “<Object name> has been submitted for your approval”.

- When the approval or rejection happens, the e-mail notification (based on the emailAddress parameter) contains the outcome of the action taken by the approver. The subject of the e-mail can be, for example, “Opportunity has been approved”.
You can only minimally change the way the BPM Worklist and e-mail notifications appear by appending information using some of the input parameters.

To change BPM notification, for example, let’s assume you are creating a workflow using the Opportunity object and have specified only the approvers and heldEntityStatusField parameters. When such an approval flow is triggered, the notification in BPM Worklist application shows "Opportunity has been submitted for your approval", which is a standard notification.

Now, let’s assume that you also want the opportunity name and region to appear in the notification. Assuming that the opportunity is Pinnacle Deal and the region is West, you can change the BPM notification to a more meaningful one by configuring the parameters as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
<th>Field Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>‘Matt Hooper, Manager of Created By, Resource Group: Direct Reports of Created By’</td>
<td>Specifies one or more people who receive the approval request.</td>
<td>Run Time</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>‘Approvalstatus_c’</td>
<td>Specifies the field that displays the status of the approval request.</td>
<td>Literal</td>
</tr>
<tr>
<td>emailAddress</td>
<td>[Optional] '<a href="mailto:jsmith@pinnacle.com">jsmith@pinnacle.com</a>, <a href="mailto:clopez@pinnacle.com">clopez@pinnacle.com</a>'</td>
<td>Specifies e-mail addresses of one or more individuals who receive the approval or rejection notification.</td>
<td>Literal</td>
</tr>
<tr>
<td>title1</td>
<td>[Optional] Name</td>
<td>Appends the name of the opportunity to the title.</td>
<td>Run time</td>
</tr>
<tr>
<td>title2</td>
<td>[Optional] Region</td>
<td>Appends the name of the region (of the opportunity) to the title.</td>
<td>Run time</td>
</tr>
</tbody>
</table>

Now trigger the approval flow again. This time the notification in the BPM Worklist application shows "Opportunity Pinnacle Deal West has been submitted for your approval," which follows the format "<Object name> <title1> <title2> has been submitted for your approval".

Similarly, when an approver takes an action on the request, the title1 and title2 parameters are also appended to the e-mail notification. For example, when the request is approved, the subject of the e-mail shows "Opportunity Pinnacle Deal West has been approved".

To display the notifications in a different format, you can specify only some of the optional parameters.

### Specifying the Trigger Condition in Object Workflows

You must ensure that the trigger condition for the object workflow contains the field that you update to trigger the workflow. For example, if you have defined your own field Submit_c of type check box, the trigger condition looks like: `isAttributeChanged('Submit_c') & Submit_c=='Y'`.

Defining an expression prevents the triggering of the object workflow each time an update or create event point occurs.

For more information on how to set trigger conditions, see Configuring Object Workflows: Example topic.
If you are specifying a drop down, fixed-choice list field for capturing approval status, use the lookup type ZCX_HOLD_STATUS to configure the status values as per your business requirement. The default values provided in this lookup type are: APPROVALPENDING, APPROVED, and REJECTED. Ensure that ZCX_HOLD_STATUS contains the default values or the values that you have specified in this lookup type for tracking the approval status.

Note: To capture the approval status, you can use only fields of type text or fixed choice list.

Setting Up Serial Group Approval: Explained

You can create an approval workflow that requires approval from a group of people, in sequential order. This is called serial group approval. In serial approval, a list of approvers must approve a process in order (that is, the second person on the list cannot approve until the first person has done so). You manage this using Oracle Business Process Manager (BPM) and object workflows.

This topic describes the following:

- Creating the group of approvers.
- Creating the approval flow.
- Deploying the project.
- Including the approval flow in an object workflow.

For more information about human tasks, refer to the “Designing Human Tasks” chapter in the Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite.

Creating the Group of Approvers

To set up serial group approval, you must first create the group for your approvers.

1. Sign in to the BPM worklist application.
2. Navigate to Administration, Approval groups.
3. Create a group named CRMCommonApprovalGroup. Note that you must give the group this name.
4. Add the members of the group (the people you want to designate as approvers).
5. The group can be static or dynamic.
6. Save the group.

Creating the Approval Flow

Next, you create the approval flow, which is based on the standard approval flow template ExtnBusinessProcessComposite.

1. Click New on the Business Processes page.
2. Select ExtnBusinessProcessComposite as the base template.
3. Remove the standard human task step named SerialApproval.
4. From the component palette on the right side of BPM Composer, drag an interactive task step into the approval flow in the same place where you removed the existing SerialApproval step.
5. Rename the new task step (for example, `GroupApproval`).

6. Right-click the human task step and click **Implement Task**.

7. Search for all patterns and select the `SerialApprovalGroupApproval` task.

8. Click **Apply changes**.

9. Right-click the human task step again and click **Data Association**.

10. Map the inputs and outputs as shown in the table below.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>GroupApproval</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskTitle</td>
<td>title</td>
<td></td>
</tr>
<tr>
<td>taskOwner</td>
<td>taskOwner</td>
<td></td>
</tr>
</tbody>
</table>
### Inputs

<table>
<thead>
<tr>
<th>taskOwner</th>
<th>errorAssignee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>outcome</th>
<th>taskOutcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11. Click **Apply changes**.
12. Save the process.

### Deploying the Project

Finally, you deploy the project.

1. From the main menu, select **Deployment**, **Deploy Project**.
2. In the **Deploy Project** dialog box, enter the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision ID</td>
<td>Provide an identifiable ID.</td>
</tr>
<tr>
<td>Deployer User Name</td>
<td>Your sign-in name for Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Password</td>
<td>Your Oracle Sales Cloud password</td>
</tr>
<tr>
<td>Mark composite revision as</td>
<td>Select this option.</td>
</tr>
<tr>
<td>default</td>
<td></td>
</tr>
</tbody>
</table>

3. Click **Deploy**.

A confirmation message appears after the project has been deployed successfully.
Including the Approval Flow in an Object Workflow

Now you can create an approval workflow that includes a serial list of approvers. To do this, create an object workflow that uses the **GroupApproval** flow that you created above.

1. Ensure you’re in an active sandbox.
2. In Application Composer, click **Object Workflows**.
3. Enter a name for the object workflow, and select an object (for example, Opportunity).
4. Click **Create**.
5. Click the **Create** icon next to **Business Process Flow**.
6. Enter a name for the flow.
7. Click **Search** next to **Project Name**.
8. Select **GroupApproval** from the list and click **OK**.
9. Click **Save**.

When the object workflow is invoked, the first member of the group receives the approval notification. After the first person approves the flow, it is routed to the second person in the group and so on.

The flow is considered complete when all the members of the approval group approve the request. If any member of the group rejects the request, then the record's status is changed to Rejected.

Setting Up Parallel Approval: Explained

You can create an approval workflow that requires approval from a group of people, in any order. This is called parallel group approval. In parallel approval, all members of a list of approvers must approve a process in any order (that is, the order of approvers in the list is not relevant, but all approvers must approve it). You manage this using Oracle Business Process Manager (BPM) and object workflows.

This topic describes the following:

- Creating the group of approvers.
- Creating the approval flow.
- Deploying the project.
- Including the approval flow in an object workflow.

For more information about human tasks, refer to the "Designing Human Tasks" chapter in the Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite.

Creating the Group of Approvers

To set up parallel group approval, you must first create the group for your approvers.

1. Sign in to the BPM worklist application.
2. Navigate to **Administration, Approval groups**.
3. Create a group named **CRMCommonApprovalGroup**. Note that you must give the group this name.
4. Add the members of the group (the people you want to designate as approvers).
5. Save the group.

Creating the Approval Flow

Next, you create the approval flow, which is based on the standard approval flow template ExtnBusinessProcessComposite.

1. Click **New** on the Business Processes page.
2. Select **ExtnBusinessProcessComposite** as the base template.
3. Remove the standard human task step named **SerialApproval**.
4. From the component palette on the right side of BPM Composer, drag an interactive task step into the approval flow in the same place where you removed the existing **SerialApproval** step.
5. Rename the new task step (for example, **ParallelApproval**).
6. Right-click the human task step and click **Implement Task**.
7. At the bottom of the screen, search for all patterns and select the **ParallelApprovalGroupApproval** task.

8. Click **Apply changes**.
9. Right-click the human task step again and click **Data Association**.
10. Map the inputs and outputs as shown in the table below.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>GroupApproval</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskTitle</td>
<td>title</td>
<td></td>
</tr>
<tr>
<td>taskOwner</td>
<td>taskOwner</td>
<td></td>
</tr>
<tr>
<td>taskOwner</td>
<td>errorAssignee</td>
<td></td>
</tr>
<tr>
<td>&quot;REJECT&quot;</td>
<td>defaultOutcome</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>VotingPercentage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>outcome</td>
<td>taskOutcome</td>
</tr>
</tbody>
</table>
In parallel approval, you must pay attention to the parameter votingPercentage. This parameter determines how many group members need to approve the request for the record to be considered approved. By setting the parameter to be 100, you specify that every approver in the list must approve the request.

11. Click Apply changes.
12. Save the process.

Deploying the Project
Finally, you deploy the project.

1. From the main menu, select Deployment, Deploy Project.
2. In the Deploy Project dialog box, enter the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision ID</td>
<td>Provide an identifiable ID.</td>
</tr>
<tr>
<td>Deployer User Name</td>
<td>Your sign-in name for Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Password</td>
<td>Your Oracle Sales Cloud password</td>
</tr>
<tr>
<td>Mark composite revision as default</td>
<td>Select this option.</td>
</tr>
</tbody>
</table>

3. Click Deploy.

A confirmation message appears after the project has been deployed successfully.

Including the Approval Flow in an Object Workflow
Now you can create an approval workflow that includes a parallel list of approvers. To do this, create an object workflow that uses the ParallelApproval flow that you created above.

1. Ensure you’re in an active sandbox.
2. In Application Composer, click Object Workflows.
3. Enter a name for the object workflow, and select an object (for example, Opportunity).
4. Click Create.
5. Click the Create icon next to Business Process Flow.
6. Enter a name for the flow.
7. Click Search next to Project Name.
8. Select GroupApproval from the list and click OK.
9. Click Save.

When the object workflow is triggered, all the users that are part of CrmCommonApprovalGroup will see the approval request. Based on the voting percentage, the outcome of the approval task is determined. In this example, because we mapped votingPercentage to 100, all the members of the group must approve the task before the record is finally approved.

The flow is considered complete when the specified percentage of members of the approval group (in this case, 100%) approve the request. If any member of the group rejects the request, then the record’s status is changed to Rejected.

Setting Up Supervisor Hierarchy Approval: Explained

You can create an approval workflow that requires approval from a hierarchical group of approvers. This is called supervisory hierarchy approval. You manage this using Oracle Business Process Manager (BPM) and object workflows.

This topic describes the following:

- Creating the approval flow.
- Deploying the project.
- Including the approval flow in an object workflow.

For more information about human tasks, refer to the "Designing Human Tasks" chapter in the Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite.

Creating the Approval Flow

Next, you will create the approval flow, which is based on the standard approval flow template ExtnBusinessProcessComposite.

1. Click New on the Business Processes page.
2. Select ExtnBusinessProcessComposite as the base template.
3. Remove the standard human task step named SerialApproval.
4. From the component palette on the right side of BPM Composer, drag an interactive task step into the approval flow in the same place where you removed the existing SerialApproval step.
5. Rename the new task step (for example, SupervisorHierarchyApproval).
6. Right-click the human task step and click Implement Task.
7. At the bottom of the screen, search for all patterns and select the SerialSupervisorHierarchyApproval task.
8. Click Apply changes.
9. Right-click the human task step again and click Data Association.
10. Map the inputs and outputs as shown in the table below.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>GroupApproval</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskTitle</td>
<td>title</td>
<td></td>
</tr>
<tr>
<td>taskOwner</td>
<td>taskOwner</td>
<td></td>
</tr>
<tr>
<td>taskErrorAssignee</td>
<td>errorAssignee</td>
<td></td>
</tr>
<tr>
<td>Inputs</td>
<td>GroupApproval</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>Integer representing the number of levels in the hierarchy (for example, 4)</td>
<td>noOfLevels</td>
<td></td>
</tr>
<tr>
<td>User name of user at top of hierarchy (for example, 'lisa. jones')</td>
<td>startingParticipant</td>
<td></td>
</tr>
<tr>
<td>outcome</td>
<td></td>
<td>taskOutcome</td>
</tr>
</tbody>
</table>

In the inputs to the Serial Supervisor Hierarchy Approval human task, you can provide the starting approver, up-to approver, or number of levels.

11. Click **Apply changes**.
12. Save the process.

**Deploying the Project**

Finally, you will deploy the project.

1. From the main menu, select **Deployment, Deploy Project**.
2. In the **Deploy Project** dialog box, enter the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision ID</td>
<td>Provide an identifiable ID.</td>
</tr>
<tr>
<td>Deployer User Name</td>
<td>Your sign-in name for Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Password</td>
<td>Your Oracle Sales Cloud password</td>
</tr>
<tr>
<td>Mark composite revision as default</td>
<td>Select this option.</td>
</tr>
</tbody>
</table>

3. Click **Deploy**.

A confirmation message appears after the project has been deployed successfully.
Including the Approval Flow in an Object Workflow

Now you can create an approval workflow that includes a parallel list of approvers. To do this, create an object workflow that uses the ParallelApproval flow that you created above.

1. Ensure you're in an active sandbox.
2. In Application Composer, click Object Workflows.
3. Enter a name for the object workflow, and select an object (for example, Opportunity).
4. Click Create.
5. Click the Create icon next to Business Process Flow.
6. Enter a name for the flow.
7. Click Search next to Project Name.
8. Select SerialSupervisorHierarchyApproval from the list and click OK.
9. Click Save.

When the object workflow is triggered, the starting approver (in our example, lisa.jones) receives the approval notification. When lisa.jones approves, the request is routed to her manager, and so on until the specified number of levels (in our example, 4) are reached. When approvers at all specified levels approve the request, the record is considered to be approved.

Multiple Approvals on a Single Record: Explained

You can set up a record to request a single approval or multiple approvals. If you request multiple approvals, you can do it in one of two ways:

- Single approval status field
- Multiple approval status fields (one for each approval)

Single Approval Status Field for Multiple Approvals

If you use a single approval status field (for example, one named “Approval Status”), then each stage of the approval process is requested individually, one after the other. After each approval is requested and obtained, the field is set to Approved. When the record is resubmitted for the next approval, the field is set to Pending Approval and the process is repeated until all required approvals are obtained.

Using Multiple Approval Status Fields for Multiple Approvals

The best practice is to use multiple approval status fields, which makes it easier to differentiate each approval on the record. This is especially true if some business logic depends on each approval.

Multiple approvals are processed in serial fashion; you can submit a new approval only after the previous approval process on the record has been completed.

Here’s an example of how you might set up multiple approval status fields.

1. Start by creating a fixed choice list (for example, Stage Status) including each record that will need approval. In this example, these are:
   - Technical Proposal
   - Financial Proposal
   - Contract
2. Next, create a fixed-choice list including each approval status (you can use the standard list values from ZCX_HOLD_STATUS):
   - Technical Approval Status
   - Financial Approval Status
   - Contract Approval Status


4. Create three workflows with the following criteria:
   - heldEntityStatusField = 'TechnicalApprovalStatus_c' OWF criteria: isAttributeChanged('StageStatus') && StageStatus = 'Technical Proposal'
   - heldEntityStatusField = 'TechnicalApprovalStatus_c' OWF criteria: isAttributeChanged('StageStatus') && StageStatus = 'Technical Proposal'
   - heldEntityStatusField = 'ContractApprovalStatus_c' OWF criteria: isAttributeChanged('StageStatus') && and StageStatus = 'Contract'

For the first approval, the Sales officer prepares the Technical Proposal document with revenue lines and sends it for manager approval. After it is approved, the Technical Approval Status field is changed to Approved. The same document is submitted to the customer for signature.

For the second approval, the sales officer prepares the Financial Proposal document (which lists payment terms) and submits it for approval by the Regional Manager and the CEO. After it is approved, the Financial Approval Status field is changed to Approved. The same document is submitted to the customer for signature.

For the third approval, after several stages in Opportunity, the Contract document is prepared and sent for approval by the CEO and Regional Manager. After it is approved, the Contract Approval Status field is changed to Approved. The document is submitted to customer for contract activation. If the customer rejects the terms, revise the contract and repeat the third approval process.

Error Behavior

If multiple approvals are submitted in parallel, then error notifications should be sent to the opportunity owner, who can manually ensure that the approval is resubmitted after the current approval is completed.

Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example

This topic contains an end-to-end example showing how you configure an object workflow to trigger approval requests for creating sales leads.

You are a sales administrator and your management has asked you to create a workflow to automate the business process as follows:

1. An approval process should be triggered when a sales representative creates a sales lead with a deal amount greater than USD 100000.
2. When the approval process is triggered, the sales manager of that representative should automatically receive a request for approving the deal.

You use three applications and two roles to configure and test the automated flows; therefore, for ease of understanding, the end-to-end procedure has been logically organized under the following steps:

1. Creating a field to track the approval status.
You use Application Composer to create a custom field to track the status of approvals.

2. Publishing the sandbox and verifying the addition of the field.
   You publish the sandbox to bring the custom field into the mainline metadata.

3. Creating and deploying an approval flow.
   You use the Oracle Business Process Composer to create and deploy an approval flow.

4. Creating object workflow to trigger the approval flow.
   You create an object workflow using the Sales Lead object and set the trigger condition on the Deal Amount field.

5. Triggering and testing the object workflow and approval flow.
   You trigger the object workflow, test the triggering of the associated approval flow, approve the request using Worklist, and then verify the approval.

Step 1: Creating a Field to Track the Status of Approvals

In this step, you create a field to track the status of the approval and add this field to the desired pages.

1. Sign in using Administrator role.
2. Create a sandbox and activate it.
   For more information on sandboxes, see Sandboxes: Explained topic.
3. Navigate to Application Composer.
4. In the left pane, expand Standard Objects.
5. Expand Sales Lead.
6. Click Fields.
   Fields page opens.
7. Under the Custom tab, select Action > Create.
   Select Field Type dialog box appears.
8. Select Text.
9. Click OK.
   Create Text Field page opens.
10. In the Appearance region, enter Large Deal Approval in the Display Label field.
11. Click Save and Close.
12. In the left pane, select Pages under Sales Lead.
   Sales Lead: Pages page opens.
13. Under Desktop Pages tab, click Create Lead.
   Create Lead page opens.
14. In the Configure Detail Form region, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
15. Click Save and Close.
   You have now added a custom approval field to the Create Lead page. You are back to Sales Lead: Pages page.
16. Under Desktop Pages tab, click Show Details.
   Show Details page opens.
17. In the Configure Detail Form region, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
18. Click Save and Close.
You have now added a custom approval field to the Show Details page.

Note: You are currently working within your sandbox. You have created a custom field and have added that field to two pages. Configurations done within a sandbox for object workflows are not available to the mainline metadata unless published.

Next, publish your sandbox to make your changes available for use in the mainline metadata.

Step 2: Publishing the Sandbox and Verifying the Addition of the Field

In this step, you publish your sandbox and verify that the Large Deal Approval field has been added to Create Lead and Show Details pages.

For more information on creating and managing sandboxes, and the considerations involved when working with sandboxes, see Sandboxes: Explained topic and its related topics.

To publish the sandbox and verify the addition of the fields:

1. In the global region, click on your user name to open Settings and Actions menu.
2. Select Manage Sandboxes.
3. Select your sandbox and click Publish.
4. Click the Navigator menu at the top.
5. Click Lead Qualification under Sales.
6. Click Create Lead.

Create Lead page opens.

7. Verify that Large Deal Approval field has been added to the Create Lead page.
8. Click Cancel. You create a deal later in this example.

Step 3: Creating and Deploying an Approval Flow

In this step, you create and deploy a business process (also called project) for approval flow using Oracle Business Process Composer.

For more information on working with business process flows, see related topics Object Workflows and Business Processes: Critical Choices and Object Workflows and Business Processes: How They Work Together.

To create and deploy a process using Oracle Business Process Composer:

1. Navigate to Application Composer.
2. Select Business Processes.

Business Processes page opens.

3. Select Actions > Create.

Create Business Process Flow dialog box appears.

4. Enter Sales Lead Approval in the Name field.
5. Click OK.

Oracle Business Process Composer opens in a separate Window.
Note: If a blocker is enabled on your browser, the business process composer may not open after you click OK, and your browser may get locked. If that happens, use the browser back button to go back to the Business Processes option in the application composer, and click Edit for the business process (approval flow) that you were creating.

For this example, you are using a seeded template called ExtnBusinessProcess on as-is basis. You can edit this template to suit your business case. For more information on this template, see Object Workflows and Business Processes: Critical Choices topic.

This figure shows the seeded template.

6. Click **Deploy Project** using the main menu.

This figure shows the main menu.
Deploy Project dialog box appears.

7. In the Deploy Project dialog box, enter the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision ID</td>
<td>Provide an identifiable ID.</td>
</tr>
<tr>
<td>Deployer Name</td>
<td>Your sign in name for Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Password</td>
<td>Your password for Oracle Sales Cloud.</td>
</tr>
<tr>
<td>Mark composite revision as default</td>
<td>Select this option.</td>
</tr>
</tbody>
</table>

8. Click **Deploy**.

A confirmation message appears after your project has been deployed successfully.


10. Confirm as follows whether the project has been deployed successfully:

   a. On the Overview page of the Application Composer, click **Business Processes**.

   Business Processes page opens.

   b. A green check-mark appears under the **Deployed** column for the project you created.

You have now successfully deployed a project with the name Sales Lead Approval.

Next, configure an object workflow to call the Sales Lead Approval project.

**Step 4: Creating Object Workflow to Trigger Approval Flow**

In this step, you configure an object workflow with the required trigger condition and specify the inputs to pass from object workflow to the Sales Lead Approval process.

To configure an object workflow:

1. Navigate to Application Composer.
2. On the Overview page, click **Object Workflows**.

   Object Workflow page opens.

3. Select **Actions > Create**.

   Create Object Workflow page opens.

4. Select **Sales Lead** from the **Object** list.
5. Enter **Deal amount more than USD 100000** in the **Name** field.
6. In the Event Point and Condition region:
   a. Select **When a record is created** option.
   b. Click the **Expression Builder** icon.

   Expression Builder opens.

   Enter `if(DealAmount>100000 && CurrencyCode=='USD') return true;`

   c. Click **OK**.

8. Configure the event action as follows:

   a. Enter **Sales Lead Approval** in the **Name** field.
   b. In the **Project Name** field, search and select **Sales Lead Approval** project that you had created and deployed.

This figure shows how you search and select a deployed business process (approval flow).

   ![Business Process Flow Details](image)

   ![Search and Select Business Process Flow](image)

   c. Click **OK**.
   d. In the Inputs region, specify the input parameters that you want to pass to the deployed approval flow using the **Select Default Value** button. You can also type-in these parameters.

   For more information on input parameters, see Object Workflows and Business Processes: Critical Choices topic. You must mark these values as **Literal** to pass the values as is. Literal values appear enclosed in single quotes.

   ![Note:](image)

   In this example, we are mapping only the mandatory parameters. For more information on the available input parameters in the default template, see Object Workflows and Business Processes: Critical Choices topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description of the Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>‘SALES MANAGER’</td>
<td>One or more recipients of the approval request.</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>‘LargeDealApproval_c’</td>
<td>The field that holds the status of the approval.</td>
</tr>
</tbody>
</table>

9. Click **Save** to save the event action.

10. Click **Save** to save the object workflow.

You have now configured the object workflow.

Next, test whether the object workflow calls the approval flow when the trigger condition is satisfied...
Step 5: Triggering and Testing the Object Workflow and Approval Flow

In this step, first trigger the object workflow and then test whether the approval flow runs fine.

1. Sign in using Sales Representative role.
2. Click the Navigator menu at the top.
3. Click Lead Qualification under Sales.
4. Click Create Lead.

Create Lead page opens.

5. Enter New Deal in the Name field.
6. In the Deal Size field, select USD and enter 120000.
7. Click Save and Close.

Creating a lead with deal size (or amount) of more than USD 100000 triggers the object workflow, because you had set the trigger condition for object workflow as: `if(DealAmount>100000 && CurrencyCode=='USD') return true;`

To verify whether the workflow has been triggered, check the status in the approval field as follows:

   a. Edit New Deal lead.
   b. Note the status in the Large Deal Approval field, which is set to APPROVALPENDING by default.

You are currently signed in using Sales Representative role. You must first sign out and then sign in using Sales Manager role to approve the request.

8. Sign out, and then sign in using Sales Manager role.

A new task or request appears on the Home page under the Worklist region.

9. Approve the request.

You are currently signed in as a Sales Manager. You must first sign out and then sign in as a Sales Representative to verify whether the status in the Large Deal Approval field has changed.

10. Sign out, and then sign in using Sales Representative role.
11. Click the Navigator menu at the top.
12. Click Lead Qualification under Sales.

Overview page opens.

13. Under Leads tab, click New Deal.

Edit Lead: New Deal page opens.

14. Verify that the status in the Large Deal Approval field has been set to APPROVED.

You have now successfully triggered the object workflow to run the associated approval flow.

Related Topics
   • Sandboxes: Explained

Configuring an Approval Process for Opportunities: Worked Example

This topic outlines the steps required to set up approvals for opportunities using a business process flow provided by Oracle.
The approval process outlined in the following figure is triggered when a condition you enter as a Groovy script is true. In this example, the process is triggered when a salesperson enters an opportunity revenue amount greater than USD 100,000.

After the process is triggered:

1. The approval process sends approval requests to one or more users.

   In this example, the notification is sent to one user. If you specify multiple users, then all users must approve before the opportunity is considered approved.

   You can enter identifying information into the subject line of the approval notification to help approvers understand what they are approving. In this example, you enter the opportunity name and the account. The body of the notification includes the name of the person who updated the opportunity.

2. The approval process updates the approval status to PENDING APPROVAL to indicate that an approval request was sent.

   In this example, you create a custom field to display the status on the Edit Opportunity page.

3. After the approver views the notification and clicks the Approve or the Reject button, the process updates the status to APPROVED or REJECTED and sends a confirmation back to the user who updated the opportunity.
4. Optionally, the process can send an informational e-mail about the process outcome to the recipients of your choice.

You create the approval process using Application Composer while working in a sandbox, so you can test the process before publishing it. To create the approval process, you:

1. Create a custom text field to display the approval status.
2. Add the custom field to the Edit Opportunity page.
3. Create an object workflow that triggers the approvals business process provided by Oracle and supplies the required inputs, including the name of the user who receives the approval notification.
4. Test the approval process before publishing it.
Creating a Custom Field to Display the Approval Status

1. Sign in as a sales administrator or a setup user.
2. Activate an existing sandbox or create a new one:
   a. Click the user name and select Manage Sandboxes from the Settings and Actions menu.
   b. If you want to create a new sandbox, click New (the plus sign icon), enter a sandbox name, and click Save and Close.
   c. Select the sandbox name and click Set as Active.

   The sandbox name appears at the top of the page.

3. Navigate to Application Composer.

   The link is available in the Navigator under the Configuration heading.

4. In the left pane, expand Standard Objects.

5. Expand Opportunity.

6. Click Fields.

   The Fields page appears.

7. On the Custom tab, select Create from the Action menu.

   The Select Field Type window appears.

8. Select Text.

9. Click OK.

   The Create Text Field page appears.

10. In the Display Label field in the Appearance region, enter Large Deal Approval.

    The application creates the technical name and API Name based on your entry. The API name is the name followed by an underscore and the letter c (_c). You enter the API name as one of the parameters of the workflow to indicate this is the field that displays the approval status.

11. Click Save and Close.

You are now ready to display the custom field on the Edit Opportunity page.

Displaying the Custom Field on the Edit Opportunity Page

1. In the Application Composer left pane, select Pages under Opportunity.

   Opportunity: Pages page appears.

2. On the Simplified Pages tab scroll down to Details Page Layouts section and edit any custom layout. If none exists, then duplicate the standard layout and edit the resulting custom layout.

3. In the Subtabs region, click Edit for the Summary.

   The Edit Summary page appears.

4. Move the Large Deal Approval field from the Available Fields box to the Selected Fields box.

5. Click Save and Close.

6. Click Done.

7. Click the Application Composer link at the top left corner of the screen to return to the Application Composer Overview page.
Creating an Object Workflow to Trigger the Approval Flow

With the sandbox still active, configure an object workflow with the trigger condition and the inputs to pass to the approvals process.

1. On the Application Composer page, make sure that Sales is still selected as the Application.
2. Click Object Workflows.

The Object Workflow page appears.
3. Select Create from the Actions menu.

The Create Object Workflows page appears.
4. Select Opportunity from the Object list.
5. Enter Deal amount more than USD 100000 in the Name field.
6. In the Event Point and Condition region:
   a. Select the When a record is updated option.
b. Click the **Groovy Builder** icon, highlighted with callout 1 in the following figure.

The Expression Builder window appears.

Enter the Groovy script:  

```
if (nvl(Revenue,0)>100000 && isAttributeChanged('Revenue')) return true;
```

You can ignore the warning.

c. Click **OK**.

7. In the Actions region, click **Create** (the plus icon on the right side of the page) for **Business Process Flow**.

The Create Action: Business Process Flow page appears.

8. Enter **Opportunity Approval** in the **Name** field.

9. In the **Project Name** field, search and select **ExtBusinessProcessComposite**. This is the name of the approval business process provided by Oracle.
10. The Inputs region lists the parameters you can pass to the business process. In the Inputs region, make the following entries:

   a. In the **approvers** field, enter the user name of the user who is to receive the approval notifications in single quotation marks. For example: *'lisa.jones'*.  
   b. In the **heldEntityStatusField**, enter the API name of the custom field you created in single quotation marks. For example: *'LargeDealApproval_c'*.  
   c. In the **emailAddress** field, enter the e-mail addresses in single quotations of people you want to be notified of the process outcome. Separate multiple e-mails with commas. For example: *'JoanneWong@nomail.com', 'tanakas@nomail.com'*.  
   d. In the **title1** and **title2** fields, enter the two fields that you want to display in the subject line of the notification:  
      
      The notification subject line appears as: **Opportunity <title1><title2> Is Submitted for Your Approval**  
      
      In this example, we are adding the opportunity name and the account fields, so make the following entries in the **Default Value** field:  
      
      * For **title1**, enter **Name**.  
      * For **title2**, enter **AccountName**.
You can view all of the fields available for insertion, by clicking **Select Default Value**, the function icon to the right of the field, highlighted by callout 1 in the following figure.

### Create Action: Business Process Flow

- **Object**: Opportunity
- **Type**: Business Process Flow
- **Name**: oppy approval

#### Execution Schedule

#### Business Process Flow Details

- **Project Name**: ExtBusinessProcessComposite

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>'lisa.jones'</td>
</tr>
<tr>
<td>heldEntityStatus</td>
<td>'LargeDealApproval_C'</td>
</tr>
<tr>
<td>emailAddress</td>
<td></td>
</tr>
<tr>
<td>title1</td>
<td>Name</td>
</tr>
<tr>
<td>title2</td>
<td>AccountName</td>
</tr>
</tbody>
</table>

To insert a field:

1. Select the field.
2. Click **Insert** to move the field to the Value region.
3. Click **OK**.

11. Click **Save and Close**.

---

**Oracle Sales Cloud**

**Extending Sales**

**Chapter 8**

**Creating Object Workflows**

---

**Oracle**

---

343
Testing the Approval Flow

To test the approval flow, you create an opportunity and enter a revenue amount greater than USD 100,000 and then sign in as the approver to view the notification and approve.

1. While signed in as the same user who performed the setup, click the Home icon to return to the springboard.
2. Double-click on the Sales icon and on the Opportunities icon to open the Opportunities work area.
3. Click Create Opportunity.

   The Create Opportunity page appears.

4. Enter a name.
5. Click Save and Continue.

   The Edit Opportunity page appears.

6. Click Add in the Products region to add a revenue line to the opportunity.

   Notice that the Large Deal Approval field you added to the page does not display a status. This is because the approval flow is not triggered yet.

7. Enter a product group or a product and an amount greater than 100000 to trigger the approval workflow.
8. Click Save and Close.

   Saving triggers the approvals process.

9. Wait a few minutes to let the approval process run and view the opportunity again. The Large Deal Approval field now displays PENDING APPROVAL.
10. Sign out by clicking the user name and the Sign Out link.
11. Sign in as the approver.
12. Click Notifications (the bell icon) at the top of the screen.

   The Recent Notifications window appears displaying the new approval notification.
13. Click the notification subject line to display the notification body and click Approve.
14. Sign out and sign in again as the administrator.
15. Click Notifications (the bell icon) at the top of the screen.

   The Recent Notifications window appears displaying the approval confirmation.
16. Navigate back to the Opportunities work area and view the opportunity you created.
17. The Large Deal Approval field displays APPROVED.
18. You must publish the Sandbox if you want to make the opportunity approval available for use.

Object Workflows and Groovy Scripts

Object Workflows and Groovy Scripts: How They Work Together

This topic explains how you use object workflows to trigger Groovy scripts. When you configure object workflows, you also specify the actions that the workflow must perform when triggered. One of the actions you configure is the Groovy Script action.

Before you begin, you might want to read Object Workflows: Explained and its related topics to familiarize yourself with what object workflows are, how you set trigger conditions, and the event actions that object workflows support.
Overview

A Groovy script action in an object workflow allows you to call an external Groovy object function that has been previously defined. For example, you might trigger a Groovy action to perform cross-object updates, perform post-processing of large object hierarchies, or asynchronously call Web services.

As a simple example, let’s assume that when you close an opportunity, you also want to close all the associated activities for that opportunity.

Using a trigger could cause a delay if the opportunity includes many associated activities, so instead you define an object workflow to trigger when the opportunity’s status is updated to Lost. The object workflow calls a Groovy object function that queries the activities related the opportunity and sets their status to "Canceled."

Note: Typically, real-world processing logic used in Groovy script actions is much more complex than the simple example described here.

You configure a Groovy Script action in object workflows by defining an object function, then creating an object workflow that, when launched at run time, executes the object function. For examples of how you configure various event actions in object workflows, see Configuring Object Workflows: Worked Example and Configuring Object Workflows to Trigger Approval Requests for Creating Sales Leads: Worked Example.

Configuring a Groovy Script Action in Object Workflows

Object workflows contain a set of actions that can be run when the workflow is triggered. One of these actions is the Groovy Script action.

To configure a Groovy Script action:

1. Navigate to Application Composer.
2. On the Overview page, click Object Workflows.
3. Select Create from the Actions menu.
   - The Create Object Workflow page opens.
4. Specify a name for your workflow.
5. Specify the trigger condition in the Event Point and Condition region.
6. Under the **Actions** region, click **Create** for Groovy Script.

The Create Action: Groovy Script page opens. You use this page to specify the Groovy object function you want to call.

This figure shows the Create Action: Groovy Script page.
7. Provide a meaningful name for your Groovy Script action.
8. Specify an **Execution Schedule**.
9. Select a **Method Name** from the drop-down list. This list displays all Groovy object functions that are visible to the action. You can also click the plus sign next to the Method Name field to display the Create Object Function page and define a new function. See Defining an Object Function for Use in Object Workflows.

The script appears in read-only form in the **Script** field.
10. Click **Save**.

### Defining an Object Function for Use in Object Workflows

If you haven’t previously defined an object function to select from the **Method** list on the **Create Action - Groovy Script** page when creating your object workflow, you can define one directly from that page. To define an object function for use with the Groovy Script action:

1. On the **Create Action - Groovy Script** page, click the plus sign next to the **Method Name** field.

   The Create Object Function page displays.

2. Enter a name for the function.
3. Select **void** in the **Returns** field.

4. Select **Callable by External Systems** in the **Visibility** field.

5. Do not specify any input parameters.

6. Enter the script in the **Edit Script** field.

7. Click the **Validate** button to validate the script, and fix any errors.

8. Click **Save and Close**.

Your object function is now ready to be called by a Groovy Script action in your object workflow. You can select it from the **Method** list when configuring an object workflow’s Groovy Script action.

### Configuring Object Workflows: Worked Example

This example demonstrates how to create an object workflow and set a trigger condition for invoking (or launching) the workflow. The trigger condition that will launch the workflow will be when the budget availability date for an opportunity is updated.

As part of the example, you configure these three event actions:

- **Field Updates**: Set the opportunity Close Date to 7 days from the new budget availability date. Additionally, lower the Level of Risk for the opportunity and set the Strategic Value to Medium.
- **Task Creation**: Create a task for the opportunity owner to follow up with the customer.
• E-mail notification: First, create an e-mail template for sending e-mail notifications. Then, notify the entire opportunity team about the change in the budget availability date using e-mail.

This example has been split into the following steps:

1. Creating a workflow and setting its trigger condition.
2. Configuring a Field Updates event action.
3. Configuring a Task Creation event action.
4. Creating an E-Mail Template and then creating an E-Mail Notification event action.
5. Creating an Opportunity record.
6. Editing the Opportunity Record to Trigger the Workflow and Verifying the Invoked Event Actions.

1. Creating an Object Workflow and Setting its Trigger Condition

In this example, you create an object workflow using an Opportunity object and create a trigger condition for the workflow using Groovy script. When the budget availability date is changed, the workflow will be triggered.

1. Click the Navigator menu.
2. Click the Application Composer link.

**Note:** You might need to click the more >> link if you don’t see Application Composer.

The Overview page of Application Composer appears. This page shows the various tasks available to modify and extend your Oracle Sales Cloud application. In this activity, you are configuring a workflow for the Opportunity object.

3. In the Overview region, click the Object Workflows link.

The Object Workflows page appears. You can use this page to search for an existing workflow or create a workflow. In this activity, you create a workflow.

4. Click the Actions menu below the Search region.
5. Click the Create menu item.

The Create Object Workflow page appears.

6. You must first select an object for which you are creating a workflow. Click the Object list.
7. Click the Opportunity list item.
8. Enter "Budget Date Revised" in the Name field.
9. In the Event Point and Condition region, click the When a record is updated option.
10. You set the trigger condition using Groovy script. Launch the expression builder to create the condition. In the Event Point and Condition region, click the expression builder button to open the expression builder.

The Expression Builder dialog appears.

11. Before you proceed, ensure that the Fields tab is selected.
12. Identify the correct application programming interface (API) name for the field you want to use for defining your trigger condition. Under the Display Label column in the Opportunity: Fields table, locate and click the Date Budget Available cell.
13. Insert Date Budget Available into the expression builder. Click the Insert button.
14. Enter your script in the Expression area using the BudgetDateAvailable field you just inserted. Write a script that meets all of these conditions:
   a. Only the BudgetAvailableDate is updated.
   b. The BudgetAvailableDate is not null.
   c. The opportunity record that you create is updated.
The following script has been written for you using the BudgetAvailableDate field:

```javascript
if(isAttributeChanged('BudgetAvailableDate') && 'BudgetAvailableDate' != null && contains(Name, ' 50 Solar Green Servers')) { return true; } else { return false; }
```

15. Validate your script. Click the Validate button.
16. Click the OK button.
17. You have set the event point and trigger condition for your object workflow. You now create a Field Updates event action. In this event action, you specify the values with which the Close Date, Level of Risk, and Strategic Value values will be replaced when the field updates action is triggered.

2. Configure a Field Update Event Action

Continuing from the previous step, you are on the Create Object Workflow page. In this step, you configure Field Updates event action for the object workflow and set new values for fields when the workflow is triggered.

- Set the Close Date to 7 days from the new budget availability date.
- Lower the Level of Risk from High to Low.
- Set the Strategic Value to Medium.

1. In the Actions region, click the Create button on the right of the Field Updates action.
   The Create Action: Field Updates page appears.
2. Enter “Update Close Date” in the Name field.
3. In the Execution Schedule region, keep the default setting, which is to update fields right away when the workflow is triggered.
4. In the Field Update Details region, click the Field to Update list.
5. Click the Close Date list item.
6. Click the Value list.
7. Click the Date Budget Available list item.
8. Make sure the ‘+’ operator is selected, and enter “7” in the Days field.
9. Click the Update More Fields link.
10. Click the Field to Update list.
11. Click the Level of Risk list item.
   You will lower the Level of Risk for an opportunity when the workflow is triggered. The values in the Level of Risk field are in a descending order of High, Low, and None; therefore, when you select the Populate with next value in list option for Level of Risk, the risk level will be lowered by one step. For example, if the initial value is High, it will change to Low when the workflow is invoked, because Low is the next value in the list.
12. Click the Populate with next value in list option.
13. Click the Update More Fields link.
14. Click the Field to Update list.
15. Click the Strategic Value list item.
16. Click the list on the right of the Value group of options.
17. Click the Medium list item.
18. Save the Field Update event action. In the upper-right region of the page, click the Save button.

You’re back to the Create Object Workflow page. Now you create a Task Creation event action for the sales team to follow up with the customer.
3. Configuring a Task Creation Event Action

Continuing from the previous step, you are on the Create Object Workflow page. In this example, you configure a Task Creation event action for an object workflow. A task will be created for the opportunity owner to follow up with the customer.

1. In the Actions region, click the Create button on the right of the Task Creation event action.
2. Enter “Follow Up Call” in the Name field.
3. In the Task Details region, enter “Follow up with customer on budget available date” in the Subject field.
4. You now insert a field token in the Subject field. Place your cursor where you want the token to appear. In the Subject field, click after “customer”.
5. Click the field-token list on the right of the Subject field.
6. Click the Customer list item.
7. Click the << Insert button.

Notice that the [${TargetPartyName}] token is inserted in the Subject where you placed your cursor. You need to adjust trailing or leading text-spaces as required.

8. Enter “Opportunity Budget Available Date is changed. Follow up within three days of the new available date. Primary Contact Email ID: ” in the Description field.
9. Use the field-token list on the right of the Description field to insert the following tokens one by one into the description text:
   - Opportunity Name ([${Name}])
   - Primary Contact ([${PrimaryContactPartyName}])
   - Primary Contact E-Mail ([${PrimaryContactEmailAddress}])
10. Click after “Opportunity” in the Description text.
11. Click the field-token list on the right of the Description field.
12. Click the Opportunity Name list item.
13. Click the <<Insert button.
14. Click after “Follow up with” in the Description text.
15. Click the field-token list on the right of the Description field.
16. Click the Primary Contact list item.
17. Click the <<Insert button.
18. Click after “Email ID:” in the Description text.
19. Click the Primary Contact E-Mail list item.
20. Click the <<Insert button.
21. You will now set the task’s Due Date to be 3 days after the new Budget Available Date. Click the Due Date list.
22. Click the Date Budget Available list item.
23. Make sure the plus sign + operator is selected, and use the increment button (up arrow) to set the value in the Days field to 3.
24. Click the Start Date list.
25. Click the Date Budget Available list item. Accept the default 0 (zero) in the Days field.
26. Click the Owner list.
27. Click the OwnerResourcePartyid list item.
28. Click the Assignees list.
29. Click the OpportunityResource Team list item.
30. Click the Category list, then click the Call, outbound list item.
31. Click the **Priority** list, then click the **1 - Very high** list item.
32. Save the **Task Creation** event action. In the upper-right region of the page, click the **Save** button.
33. You are back to the Create Object Workflow page. In the upper-right region of the page, click the **Save** button. This saves the object workflow.

You now create the E-Mail Notification event action. Before you create this event action, you must first create an E-Mail Template that you use for sending e-mail notifications.

4. Creating an E-Mail Template and Configuring an E-Mail Notification Event Action

Continuing from the previous step, you are now on the Object Workflows page. In this step, you create an e-mail template, which you use for sending e-mail notifications.

1. In the Common Setup pane on the left, click the **E-Mail Templates** link.

   You are on the E-Mail Templates page. You can use this page either to search and edit an existing template or to create a fresh template. In this activity, you create a template.
2. In the Search Results region, click the **Actions** menu.
3. Click the **Create** menu item.

   The Create E-Mail Template page appears.
4. Click the **Object** list.
5. Click the **Opportunity** list item.
6. Enter Budget Available Date Update" in the **Name** field.
7. Enter "Opportunity customer budget available date has changed" in the **E-Mail Subject** field.
8. You now insert a field token in the **E-Mail Subject** field. Place your cursor where you want the token to appear. Click after "Opportunity" in the **E-Mail Subject** field.
9. Click the field-token list on the right of the **E-Mail Subject** field.
10. Click the **Opportunity Name** list item.
11. Click the **<<Insert** button.
12. In the E-Mail Body region, enter "- budget available date regarding opportunity has moved to . The new opportunity close date is ." in the **E-Mail Body** field.
13. In the E-Mail Body region, use the **Fields** list item of the **Select** list to insert the following field tokens:

<table>
<thead>
<tr>
<th>Field</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>[$PartyUniqueName$]</td>
</tr>
<tr>
<td>Opportunity Name</td>
<td>[$Name$]</td>
</tr>
<tr>
<td>Date Budget Available</td>
<td>[$BudgetAvailableDate$]</td>
</tr>
<tr>
<td>Close Date</td>
<td>[$EffectiveDate$]</td>
</tr>
</tbody>
</table>

**Note:** Rich text formatting is available only if your e-mail account supports HTML format e-mail.
14. Click after "- " in the e-mail body.
15. In the E-Mail Body region, click the field-token list on the left of the Insert button.
16. Click the Customer list item.
17. Click the Insert button.
18. Click after "regarding opportunity" in the e-mail body.
19. Click the field-token list.
20. Click the Opportunity Name list item.
21. Click the Insert button.
22. Click after "has moved to" in the e-mail body.
23. Click the field-token list.
24. Click the Date Budget Available list item.
25. Click the Insert button.
26. Click after "close date is" in the e-mail body.
27. Click the field-token list.
28. Click the Close Date list item.
29. Click the Insert button.
30. Save the E-Mail Template. In the upper-right region of the page, click the Save button.
   You’re returned to the E-Mail Templates page.
31. In the Common Setup pane on the left, click the Object Workflows link.
   You now create an E-mail Notification event action using the e-mail template you just created.
32. In the Search region, click the Object list.
33. Click the Opportunity list item.
34. Click the Search button.
   The Name column in the search results lists the object workflows created for the opportunity object. From this list, you identify and select the object workflow that you are creating.
35. Click the Budget_Date_Revised cell.
36. Click the Edit button.
   The Edit Object Workflow page appears. You are editing an existing object workflow to configure and add an E-Mail Notification event action.
37. In the Actions region, click the Create button on the right of the E-Mail Notification event action.
   The Create Action: E-Mail Notification page appears.
38. Enter "Notify sales team about budget available date change" in the Name field.
39. Select the e-mail template that you created. In the E-Mail Details region, click the E-Mail Template list.
40. From the list of e-mail templates, identify and select the template that you created. Click the Budget Available Date Update template.
41. Click the Recipient Type list.
42. Click the Specific e-mail addresses list item.
43. Enter the desired information into the E-Mail Addresses field.
44. Save the E-Mail Notification event action. In the upper-right region of the page, click the Save button.
   You are returned to the Edit Object Workflow page.
45. Save the object workflow. In the upper-right region of the page, click the Save button.
   You have now configured an object workflow with three event actions. This object workflow is invoked (launched) when you update the budget available date for the opportunity that you create.
5. Creating an Opportunity Record

In this section, you trigger the object workflow that you created in the previous steps by entering the budget availability date for an opportunity. You note down the details where the changes will occur after the workflow is triggered.

1. Click the **Navigator** menu.
2. Click the **Opportunities** link under **Sales**.
   
The Overview page of Opportunities appears. You use this page to create an opportunity record.
3. In the Opportunities region, click the **Create** button.
   
The Create Opportunity page appears.
4. Verify that **Close Date** is the current date, **Sales Stage** is 01-Qualification, and **Win Probability (%)** is 0 (zero).
5. In the **Name** field, enter "50 Solar Green Servers".
6. In the **Sales Account** field, enter "Solar Inc (KIRKLAND, US)".
7. Click the **Sales Account** search button adjacent to the **Sales Account** field.
   
The Search and Select: Sales Account dialog appears.
8. In the dialog, the Solar Inc (Kirkland, US) search parameter appears in the **Name** field. Click the **Search** button.
9. In the Search Results region, click the **Solar Inc (Kirkland, US)** cell.

   **Note:** Ensure that the All Sales Accounts tab is selected.

10. Click the **OK** button.
11. In the Revenue Items region, click the **Add Row** button.
12. Enter "DG 150 Green Servers" in the **Name** field.
13. Enter "50" in the **Quantity** field.
15. In the upper-right region of the page, click the **Save and Edit** button.
   
The Edit Opportunity page appears.
16. Expand the following if not already expanded:
   
   o Show More area in the Summary region.
   
   o Revenue Items region.
   
   o Activity Center region.

17. Locate the following fields and note their current values:
   
   o **Close Date** (in the Summary region).
   
   o **Strategic Value** (under Show Less area in the Summary region).
   
   o **Level of Risk** (under Show Less area in the Summary region).
   
   o **Date Budget Available** (under Show Less area in the Summary region).

   When you update the **Date Budget Available** field to trigger the workflow, the values in the **Close Date**, **Strategic Value**, and **Level of Risk** fields should change.

18. Under the Show Less area in the Summary region, click the **Level of Risk** list.
19. Click the **High** list item.
20. In the upper-right region of the page, click the **Save** button.
21. You are still on the Edit Opportunity page. In the Activity Center region, click the **Tasks** tab.

Currently there are no Tasks with your sign-in initials.

22. In the Additional Details region, click the **Opportunity Team** tab.

23. In the Team Members table, locate your sign-in username. You will now add another team member: **Mateo Lopez**.

24. In the Opportunity Team tab, click the **Add Team Members** button.

The Select and Add: Team Members dialog appears.

25. Find a team member and click the **Done** button.

26. Search and add a contact to the opportunity. This will be the primary contact for the opportunity.

27. In the Additional Details region, click the **Contacts** tab.

28. In the Contacts tab, click the **Add Row** button.

29. Enter the desired information into the **Name** field. As you type in the contact name, the contact name and e-mail ID appear.

30. Click the desired contact.

31. In the upper-right region of the page, click the **Save and Close** button.

32. You’re back to the Overview page of Opportunities. You will now edit the opportunity record you just created and update the budget availability date to trigger the event actions.

6. Editing the Opportunity Record to Trigger the Workflow and Verifying the Invoked Event Actions

Continuing from the previous step, you are on the Overview page of Opportunities. You will trigger (launch) the object workflow that you created in the previous steps by entering the budget availability date for an opportunity.

You will also verify the following after the workflow is invoked:

- Field Updates: Verify that the opportunity Close Date is set to 7 days from the new Budget Available Date. Verify that the Level of Risk is lowered for the opportunity and the Strategic Value is set to Medium.
- Task Creation: Verify that a task is created for the opportunity owner to follow up with the customer.
- E-mail notification: Verify that an e-mail notification is sent to the entire opportunity team about the change in the opportunity's Budget Available Date.

1. From the table in the Opportunities region, identify and select the opportunity record that you created. Click the **50 Solar Green Servers** link.

2. You are on the Edit Opportunity page. Before you proceed, expand the Show More area in the Summary region if not already expanded.

3. Under the Show Less area in the Summary region, click the **Date Budget Available** (calendar) button.

4. Click 9 as the date, or click any future date.

5. In the upper-right region of the page, click the **Save and Close** button.

>**Note:** You have just changed the budget availability date and saved the record, saving the changes to the database. The object workflow will be evaluated and triggered at this time, and all the event actions will be executed. The Field Updates event action always happens first followed by other event actions in no particular order. It might take a few seconds for the updates to occur.

6. In the table under the Opportunities region, identify and select the opportunity record that you created. Click the **50 Solar Green Servers** link.
You are on the Edit Opportunity page. The workflow is now triggered and the configured event actions have been invoked. You will now verify the invocation of Field Updates event action.

7. In the Summary region, Close Date is now set to 7 days after the Date Budget Available value.
8. Under the Show Less area in the Summary region, Strategic Value is now set to Medium.
9. Under the Show Less area in the Summary region, Level of Risk has now been populated with the next value in the list, from High to Low.

You have verified the invocation of the Field Updates event action. Now, you verify the invocation of the Task Creation event action.

In the Activity Center region, the Due Date filter under the Tasks tab may prevent your task from appearing in the invoked tasks list even after it is invoked. For example, a task having a due date after 10 days will not appear if the Due Date filter is set to Next 7 Days. In this activity, you use the All Tasks filter option to search for your invoked task.

10. In the Activity Center region, click the Due Date list under the Tasks tab.
11. Click the All Tasks list item.
12. In the Tasks tab, click the arrow button adjacent to the Category list.
13. In the Search results, locate the Subject that has your sign-in initials. Click the Follow up with customer Solar Inc on budget available date link.

You are on the Edit Task page. Note these details:

- The Owner of the opportunity is the owner of this task.
- The task Assignees are persons in the opportunity sales team.
- In the Description field, the field tokens that you had inserted when creating the Task Creation event action have been replaced with values at run time.

14. Click the Done button.
15. In the upper-right region of the page, click the Save and Close button.

You have verified the invocation of the Task Creation event action. Next, verify the invocation of the E-Mail Notification event action.

16. In your e-mail client, locate the Subject of your e-mail. Click the Subject link.
17. Notice that the field tokens that you inserted while configuring the e-mail notification event action have been replaced with values at run time. You have now verified the invocation of the E-Mail Notification event action.

You have now verified the invocation of all three event actions.

Deleting Unpublished Object Workflows

You can use Application Composer to delete any unpublished object workflows and their associated actions from the current sandbox. Upon deletion, the corresponding create or update events will not be raised and none of the actions will be triggered. However, deleting an unpublished object workflow will not delete the email template or object function referenced in the object workflow actions. Also, if an action has been scheduled before the object workflow is removed, the action will continue to be executed at the scheduled time.

To delete an unpublished object workflow:

1. In Application Composer, click Object Workflows under Common Setup.

The Object Workflows page appears.
2. Click the **Object** list and select the object whose workflow you want to delete.
3. Optionally, if you know the name of the workflow, enter it in the **Name** field and specify if it is active or not using the **Active** list.
4. Click **Search**.

The page displays the object workflow based on the specified search criteria.
5. Select the object workflow and click the delete icon.
6. Click **OK** on the confirmation dialog.

**FAQs for Creating Object Workflows**

**Why are some e-mail templates invalid?**

E-mail templates can become invalid if the corresponding template cannot be found in the file repository. This can happen if the template file was migrated elsewhere or was accidentally deleted. Contact your application administrator to attempt recovering the template (file) from the file repository. If the file cannot be recovered, delete the invalid e-mail template and create a fresh one.

**Can I specify multiple approvers when configuring a business process flow event action?**

Yes. You can modify the default template to design either a parallel or serial business process flow and save it as a new project. A parallel approval is where any one approval is sufficient from the approvers specified in the process flow. A serial approval is where all the approvals are required sequentially from the specified approvers.
9 Modifying Pages

Overview

Use Page Composer to change page content, layout, and more. Using other tools, you can create new pages and change UI text, themes, infolets, and so on.

For example, you can:

- Use tools such as User Interface Text to change UI text.
- Use the Appearance page to change the look and feel of the application.
- Open an infolet page in Page Composer to modify it.
- Use the Page Integration page to create and manage pages for hosting third party applications.

**Note:** You cannot use Page Composer to configure your home page. Instead, you can use other tools such as, Structure and Appearance for home page configurations.

Related Topics

- Configuring Home Page Navigation: Procedure
- Configuring Navigation and Home Page: Overview

Page Composer Overview

Prerequisites for Modifying Existing Pages: Explained

Before modifying pages, do the following tasks:

- Understand the typical workflows for working with runtime application changes.
- Verify that the page can be modified. To do so, check if either the Customize Pages or the Customize <Page Name> Pages menu item is available in the Settings and Actions menu. If no, then that means the page can't be modified.
- Confirm that your privileges are sufficient for modifying the page.
- Activate a sandbox.

Related Topics

- Setting Up Sandboxes: Procedure
- Runtime Configuration Workflow: Explained
- Context Layers: Explained
Page Composer Views: Explained

You can use either Design view or Source view for viewing and changing page content and layout in Page Composer. To open a view option, select it from the View menu at the top left corner of the page. Although both views share many common page modification features, you can use some unique features in each view.

Design View

In Design view, you see one region that shows a WYSIWYG rendering of the page and its content. Work with components directly and access their properties from the toolbar in the chrome of the individual component.

Source View

In Source view, you see two regions:

- Selection pane, showing a WYSIWYG rendering of the page and its content
- Source pane, showing a hierarchical ordering of the page components, including some components that otherwise don’t appear on the page. You can select and configure such components in Source view.

Tip: Controls on individual components are inactive in Source view, but you can click an individual component to select it.

In Source view, you can:

- Click a component in the Selection pane to highlight the component in the hierarchical list. The cursor turns to a magnifier and a blue outline appears around the component selection. You can also traverse the hierarchy and select components directly.
- Click Edit on the view header to work with components indirectly and access their properties. You can also right click the object in the hierarchy, and click Edit.

Page Component Properties: Explained

All components have configurable properties that control, or express their appearance and functionality. Many properties are common to all component types, while some properties are unique to one component type. Use the Component Properties dialog box to view the properties of a component. To open this dialog box, select the component that you want to edit, and click Edit. You can see properties of similar functions in tabs that name the category of the properties.

Note: Properties and tabs can vary from component to component.

Component Property Tabs

This table describes the tabs that you may see in a component properties dialog box.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Settings that control component aspects that are specific, or often unique to the component. For example, on a page containing a map, a component may have a parameter that provides a choice between units of measurement.</td>
</tr>
<tr>
<td>Tab</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Display Options</td>
<td>Settings that affect the chrome of a component, including:</td>
</tr>
<tr>
<td></td>
<td>• Header, header text, and border</td>
</tr>
<tr>
<td></td>
<td>• Actions menu</td>
</tr>
<tr>
<td></td>
<td>• Edit, Remove, Expand, Collapse, and other icons</td>
</tr>
<tr>
<td></td>
<td>• Tooltips</td>
</tr>
<tr>
<td></td>
<td>For example, display options on image layout components specify the image source URL and its optional link target.</td>
</tr>
<tr>
<td>Child Components</td>
<td>The list of all components contained within and under the control of the parent component, including controls for ordering the child components.</td>
</tr>
<tr>
<td>Style</td>
<td>Settings that affect the look and feel of the component chrome or the component contents. These settings override corresponding values from a parent object, such as a component, page, and application, providing an opportunity to fine-tune appearance.</td>
</tr>
<tr>
<td>Content Style</td>
<td>For example, font, color, and dimension. Some style properties may be disabled at the component level if other page or application elements (such as the skin) don’t support modification to the property.</td>
</tr>
<tr>
<td>Events</td>
<td>Events and event handlers for all the components on the current page that the current component can consume.</td>
</tr>
<tr>
<td></td>
<td>For example, an event can be selecting a check box within the current context. The code that runs and drives the result of an event, such as making another component visible, is an event handler.</td>
</tr>
</tbody>
</table>

**Working With Components in Page Modifications: Procedure**

Use the Resource Catalog to modify pages. This catalog provides a selection of task flows, portlets, and layout components. Open the Resource Catalog in Page Composer from either Design view or Source view.

Aspects of components pertinent to page modifications include:

- Opening the Resource Catalog
- Adding components
- Hiding components manually
- Hiding components programmatically

**Prerequisites**

Activate a sandbox.

**Opening the Resource Catalog**

In Design view:

1. Open the page that you want to modify in Page Composer.
2. From the existing components, select the one that you want to be the parent component.
Tip: Alternatively, use the Add Box icons (Add Box Above, Add Box Below, Add Box Left, and Add Box Right) to insert a box component. Then select the component as the parent component.

3. Click the **Add Content** button associated with the parent component.

   The Resource Catalog appears.

Adding Components

In Design view:

1. Open the Resource Catalog.
2. In the Resource Catalog, find the component that you want to add.
3. Click the **Add** icon associated with the component.
4. Cut and paste, or drag and drop the component to place it.

In Source View:

1. Select the container component in the selection pane.
2. In the Source view toolbar, click **Add Content**.
3. In the Resource Catalog, find the component you want to add.
4. Click the **Add** icon associated with the component.

Hiding Components Manually

Use the **Show Component** property to specify whether the component appears to users at runtime. By default, all components are visible. To manually hide a component, deselect **Show Component** on the Component Properties dialog box.

If the component is a:

- Child component, then deselecting the **Show Component** property hides only the child component.
- Parent component, then deselecting the **Show Component** property of the parent component hides the parent and all child components it contains. So, when you hide a parent component, you automatically hide all child components.

You can do any of the following:

- Hide a child component directly
- Hide a child component from within the parent component
- Hide a parent component and all child components

To hide a child component directly:

1. Click the **Edit** icon in the header of the child component. This opens the Component Properties dialog box.
2. Click the **Display Options** tab.
3. Deselect **Show Component**.
4. Click **OK**.

To hide a child component from within the parent component:

1. Click the **Edit** icon on the containing box’s toolbar.
2. Click the **Child Components** tab.
3. Deselect the box next to the component you want to hide.
4. Click **OK**.
To hide a parent component and all child components:

1. Click the Edit icon in the box header.
2. Click the Display Options tab.
3. Deselect Show Component.
4. Click OK.

### Hiding Components Programmatically

You can add an Expression Language (EL) expression to a component that enables you to set a condition for hiding the component. For example, suppose you have two check boxes (1 and 2) on a page. You also have a button (B) that you want to be visible only if check box 2 is selected. To step through the logic, ask yourself questions such as the ones in the following table.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Purpose of Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s the condition?</td>
<td>Check box 2 is selected</td>
<td>Determines what the occurrence, or event, is.</td>
</tr>
<tr>
<td>What action or event must happen?</td>
<td></td>
<td>Determines the component that triggers the event.</td>
</tr>
<tr>
<td>What happens when the condition is met?</td>
<td>Button “B” appears.</td>
<td>Determines the effect of the action.</td>
</tr>
<tr>
<td>What happens when the event happens?</td>
<td>Show the component: Button B (The implication is that button B is hidden until the event occurs.)</td>
<td></td>
</tr>
<tr>
<td>What property determines whether a component is visible?</td>
<td>The Show Component property</td>
<td>Determines the property the code affects.</td>
</tr>
</tbody>
</table>

So the logic is: **If 2 is checked, then the Show Component property of B is activated.**

You place the expression on the component that receives the action.

Here’s a sample code that you may add to the component.

```text
#{if checkbox2.selected = true}
```

After you think through the logic and find the correct expression, add it to the property. You can add an expression using the expression builder for the Show Component property only on dashboard pages; not on work area pages. Also, only administrators can perform this task.

To open the EL Editor and add an expression to a property for a dashboard page:

1. Click the Edit icon in the component header.
2. Click the Display Options tab.
3. Click the Edit icon next to the Show Component property, and select Expression Builder....
4. Add an expression to check for an event or condition, and set the property. Based on the result, turn the property on or off.

To hide a parent component and all child components programmatically for a dashboard page:

1. Click the Edit icon in the box header.
2. Click the Display Options tab.
3. Click the Edit icon next to the Show Component property, and select Expression Builder...
4. Add an expression to check for an event or condition, and set the property. Based on the result, turn the property on or off.

Related Topics
- Setting Up Sandboxes: Procedure

Modifying Tabs on Application Pages Using Page Composer: Worked Example

This example demonstrates how to modify tabs on pages using Page Composer.

The following table summarizes the key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In this Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who do you want to make the changes for?</td>
<td>All users</td>
</tr>
<tr>
<td>What changes are you going to make?</td>
<td>Hide the Incentive Management tab and the Projects tab from the Worklist: Notifications and Approvals page</td>
</tr>
</tbody>
</table>

Prerequisites
Activate a sandbox.

Modifying Tabs
Do the following:

1. From the Navigator menu, select Tools > Worklist.
2. Click your user image or name in the global header, and select Customize Pages in the Settings and Actions menu.
3. Select Site layer as the context layer, which affects all users.
4. In the View menu at the top of the page, make sure that the Design view is selected, which lets you navigate to the component you want to modify.
5. Click the Select tab.
6. Hover over any tab on the page, and click when the cursor turns to a magnifier and a blue outline appears around the tab.
7. Select Edit Parent Component.
8. In the Component Properties dialog box, click the Children tab.
9. Deselect Incentive Compensation and Projects to hide these tabs from the page.
10. Click OK.

Related Topics
- Setting Up Sandboxes: Procedure
Making Application Changes Visible Based on User Roles: Worked Example

This example demonstrates how you can make application changes visible to a specific user role.

To control page components conditionally based on user role:

- Create security privileges
- Add an Expression Language expression to the component property that you want to control

The following table summarizes the key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s the page object you’re securing?</td>
<td>Reports link</td>
</tr>
<tr>
<td>To which user role you want to make the changes visible?</td>
<td>Hiring managers, Sherry Callaway and Terrance Whitaker</td>
</tr>
</tbody>
</table>

Tip: Create a role just for testing application changes. Call it DEVCUST_TEST_ROLE. When you’re sure that the application change works, change the security to the appropriate role.

| Which expression to add for verifying whether a user has the appropriate privilege? | #{securityContext.userGrantedPermission['MANAGERREPORTS LINK_PRIV']} |

As a prerequisite, activate a sandbox.

Making Application Changes Visible to a User Role

Follow these steps:

1. Create a privilege.
2. Add the `MANAGER_REPORTS_LINK_PRIV` object to `DEVCAST_TEST_ROLE`.
3. Assign `DEVCAST_TEST_ROLE` to Terrance and Sherry.

   Note: Make sure that Terrance and Sherry have access to the page before you make any changes.

4. Open the page, having the Reports link that you want to modify, in Page Composer, and select the Reports link component.
5. Click the Edit icon. This opens the Component Properties dialog box.
6. Click the Display Options tab.
7. Click the Edit icon next to the Show Component property, and select Expression Builder...
8. Add an expression to verify whether the user has the appropriate privilege. Use this sample code:
   #{securityContext.userGrantedPermission['MANAGER_REPORTS_LINK_PRIV']}
9. Save the property changes, and close Page Composer.

After you change the show component property, whenever users open this page, the application evaluates the expression. Since Sherry and Terrance have the privilege, the Show Component property evaluates to be selected. Hence, Sherry and Terrance can see the Reports link while all other users can't.
Modifying Objects That Appear on Multiple Pages: Points to Consider

Use Page Composer to modify objects that appear on multiple pages. Whether your application changes affect one or more pages depends on the way you include the object on the page and the applications that you use.

Consider these points when you modify objects that appear on multiple pages.

### Appearance of Object Modifications Based on Shared and Non-Shared Task Flows

The following table briefly describes the task flow scenarios when object modifications appear on one or more pages.

<table>
<thead>
<tr>
<th>If the object is...</th>
<th>Then the modifications...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not part of a shared task flow</td>
<td>Don't appear on other pages</td>
</tr>
<tr>
<td>Part of a shared task flow</td>
<td>Do appear on all pages that include the shared task flow</td>
</tr>
</tbody>
</table>

⚠️ **Caution:** When shared task flows include embedded logic that uses data from the page, the logic can override the application changes that you make in Page Composer.

### FAQs for Page Composer Overview

**How can I move page components?**

To move page components using Page Composer:

- In Design view, drag and drop the component
- In Source view, do any of the following:
  - Cut and paste the component
  - Drag and drop the component
  - Open the Component Properties for the container component and rearrange the components on the Child Components tab

**How can I delete components from a page?**

Click the **Delete** icon in the component header in Page Composer.

⚠️ **Caution:**

- Delete a component only if you’re certain that no other components or processes depend on that component. If you’re unsure whether any dependencies exist, then hide the component instead of deleting it.
- If you delete a parent component, you delete all the child components automatically.
How can I work on application changes, but prevent users from viewing them until the changes are complete?
Create or select an appropriate sandbox, and set it as active to capture your application changes using Page Composer. When you’re ready, publish the sandbox to make your changes available to users.

Related Topics
- Managing Configurations Using Sandboxes: Explained
- Setting Up Sandboxes: Procedure
- Publishing Sandboxes: Procedure

What happens if my configurations make the page inaccessible?
You must contact an administrator, and the administrator can use the Manage Customizations task to view and, if necessary, delete your changes.

How can I reset a page or task flow to a previously saved version?
To reset a page to a previously saved version or the original ready-to-use state, click **Reset Page** in Page Composer (Design or Source view).
To reset a task flow to a previously saved version or the original ready-to-use state, click **Reset Task Flow** in Page Composer (Source view only) while you have the task flow open.

Can I edit Expression Language (EL) expressions using Page Composer?
No, Page Composer doesn’t allow editing EL expressions to protect the business logic implemented through EL expressions.

Modifying Page Content and Templates

Changing Page Layout Using Page Composer: Procedure
Page layout defines the number, placement, and orientation of content regions on a page. Although you set the layout style while creating a page, for some layouts, you can change the layout style even after adding content to the page.

> **Note:** You can’t change the page layout for all pages.

Prerequisites
Activate a sandbox.

Changing Page Layout
To change the layout of a page:
1. Open the page in Page Composer.
2. From the View menu, select **Design**.
3. Click **Change Layout**.
4. Select the new layout.
Configuring the Global Page Template: Explained

The global page template provides a common header area and the footer panel for all pages in your application. You can use Page Composer to configure the global page template. To open the global page template in Page Composer, click your user image or name in the global header, and select **Customize Global Page Template** from the Administration menu.

You can make the following changes to the global page template:

- Add components
- Edit components
  - Example: Add expression language to hide the tags link
- Delete components
  - Example: Remove the tags link

**Tip:** When you move your cursor over the global page template, the areas that you can edit display a blue outline.

You have two separate global page templates - one for the desktop user interface (UI) and the other for the simplified UI. If you click **Customize Global Page Template** from a desktop page, you will configure the template for the desktop UI. Similarly, if you click **Customize Global Page Template** from a simplified page, you will configure the template for the simplified UI. Hence, to ensure a consistent look and feel for all pages, you must configure the global page template twice, once for each UI.

### Adding Components to the Global Page Template

To add components to the global page template:

1. Open the global page template in Page Composer.
2. Select the portion of the global header to which you want to add a component, and click **Add Content**.
3. In the component catalog, select **Components** to display the list of available components.
4. Click the **Add** button associated with the component you want to add.
   - The component appears in the global header.
5. Change component properties, as appropriate. For example, if you added the Text component, enter the text that you want to display.
6. After completing your changes, click **Close**. When prompted, click **Save** to save your changes.

### Editing Components in the Global Page Template

To edit components in the global page template:

1. Open the global page template in Page Composer.
2. Select the component that you want to edit.
3. Click **Edit**.
4. Edit the component properties, and click **OK** to save your changes.
5. After completing your changes, click **Close**. When prompted, click **Save** to save your changes.
Deleting Components from the Global Page Template

To delete components from the global page template:

1. Open the global page template in Page Composer.
2. Select the component that you want to delete.
   When you move your cursor over the global page template, the areas that you can edit display a blue outline.
3. Click **Delete**. When prompted, click **Delete** to delete the component.
4. After completing your changes, click **Close**. When prompted, click **Save** to save your changes.

Editing Footers in the Global Page Template

To edit footers in the global page template:

> **Note**: For simplified pages, you can’t modify footers in the global page template.

1. Open the global page template in Page Composer.
2. Select the left region of the footer area. In the Source region, make sure the `spacer: 40px` node is selected after you select the left region of the footer area.
3. In the Source region, select the `panelGroupLayout: horizontal` node that appears above the spacer, and click **Add Content**.
4. In the component catalog, select **Components** to display the list of available components.
5. Click the **Add** button associated with the component you want to add. The component appears in the footer.
6. Change component properties, as appropriate.
7. After completing your changes, click **Close**. When prompted, click **Save** to save your changes.

For information about making skin modifications, such as selecting a different color palette, see the Oracle Fusion Applications Extensibility Guide for Developers.

Modifying Dialog Box Content Using Page Composer: Procedure

Use Page Composer and work in source view to modify the content in your dialog boxes.

**Prerequisites**

Activate a sandbox.

**Modifying Dialog Boxes**

To modify dialog box content:

1. Open the page where the dialog box appears, and then open Page Composer.
2. From the View menu, select **Source**. You must be in Source view to modify dialog box content.
3. Select the button that opens the dialog box.
4. Click **Edit** to open the Component Properties dialog box.
5. Click the **Child Components** tab.
6. Edit the dialog box content.
7. Click **Apply** to save your changes, then **OK** to save your changes and close the Component Properties dialog box.

**Related Topics**

- Setting Up Sandboxes: Procedure
Saved Search Configuration

Making Saved Searches Available to All Users: Procedure
Use Page Composer at the site layer to create and edit saved searches, and make them available for all users. Create and edit saved searches using Page Composer at the site layer. Users can run these saved searches again later to use the same search criteria and other settings. You must create or edit saved searches only at the site layer to make them available for all users.

Creating and Editing Saved Searches for All Users
Follow these steps:

1. Activate a sandbox.
2. On the search page that has a **Save** button, click your user image or name in the global header, and select **Customize <Page Name> Pages** in the **Settings and Actions** menu.
3. If prompted to select a context layer, select the site layer to open the search page in Page Composer.
4. From the View menu, select **Design**.
5. Create and edit saved searches.

> **Note:** The steps for creating and editing saved searches are the same regardless of whether you’re working on saved searches for yourself or for all users.

6. Save your changes and close Page Composer.
7. After testing your changes, publish the sandbox to make your changes available to all users.

Related Topics

- Setting Up Sandboxes: Procedure
- What gets saved when I create a saved search for searches with multiple criteria?

Saving Searches for Searches with Multiple Criteria: Procedure
On many pages, you can run a search with multiple search criteria to find specific business objects. Some of these searches have a Saved Search list, as well as a Save button after the search criteria. A saved search captures search criteria and other settings so that you can easily run the same search again later. Aside from using any predefined saved searches, you can create and edit them for your own use. If you have the right roles, you can also create and edit saved searches for other users using Page Composer.
The following figure shows an example of a search with multiple search criteria fields and a Save button. For each field, you can select an operator and enter search terms. You can also select from the Saved Search list to use an existing saved search.

```
<table>
<thead>
<tr>
<th>Search</th>
<th>Basic</th>
<th>Saved Search</th>
<th>Application Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Contains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Contains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>Reset</td>
<td>Save...</td>
<td>Add Fields</td>
</tr>
</tbody>
</table>
```

Creating Saved Searches

Follow these steps:

1. Go to a search that has a **Save** button.
2. Optionally add or reorder fields, if available.
3. Enter your search criteria values, and, click the **Save** button.
4. Name your saved search and define its settings:
   - **Set as Default:** The saved search is automatically selected whenever you open this page.
   - **Run Automatically (if available):** The saved search runs on this page as soon as you select it from the list of saved searches.

If you select both options, then the saved search automatically runs whenever you open this page.
5. Close the dialog box.

Your saved search can be limited to the current page, or in some cases available in other searches for the same object.

Changing the Search Criteria in Saved Searches

Follow these steps:

1. Select the saved search if it’s not selected already.
2. Set your search criteria, including any additional fields.
3. Click the **Save** button.
4. If the saved search is one that you created, save without changing the name.
   - If it’s predefined, then you can’t overwrite it, so you create a new saved search with a unique name.

Changing Settings or Renaming and Deleting Saved Searches

Follow these steps:

1. Go to a search that has a **Save** button.
2. Select **Personalize** from the **Saved Search** list.
3. In the Personalize Saved Searches dialog box, select a saved search.
4. Change any of the settings, where available.
   - **Set as Default:** The saved search is automatically selected whenever you open this page.
   - **Run Automatically:** The saved search runs on this page as soon as you select it from the list of saved searches.
Show in Search List: The saved search is available for you to select and run on this page.
- You can still find hidden saved searches in the Personalize Saved Searches dialog box.
- You can’t change this setting if the saved search is currently selected on your page.

Note: Some settings can’t be changed for predefined saved searches. What you do change applies only to you, unless you’re editing the saved search within Page Composer.

5. If you selected a saved search that you created, then you can rename or delete it. You can’t do so for predefined saved searches.
6. Save your changes and close the dialog box.

Related Topics
- What gets saved when I create a saved search for searches with multiple criteria?
- Creating Watchlist Items: Procedure

Saving Searches for Keyword Searches with Filters: Procedure
One type of search you might find on the page is a keyword search with filters. Some of these searches have predefined saved searches, and you can also create and edit saved searches for your own use. A saved search captures entered search terms, filters, and other settings so that you can easily run the same search again later. If you have the right roles, you can also create and edit saved searches for other users using Page Composer.

Creating and Editing Saved Searches
Follow these steps:
1. Enter search terms in the search field, and click Search, or select a saved search in the autosuggest.
2. Click the Show Filters link if filters are currently hidden.
3. Optionally organize filters as follows, depending on what’s available to you:
   - Use the Add or Reorder buttons.
   - Click the Personalize icon to show, hide, or reorder filters, and click OK.
4. Set filter values or select filters, and click the Save button.
5. Name your saved search.
   - To create a saved search, enter a new, unique name.
   - To update an existing saved search that you created, save with the original name.

You can’t overwrite predefined saved searches by using their names, but you can create a copy with a unique name.
6. Click Set as Default if you want the saved search to be automatically selected whenever you open this page.
7. Click OK.

Your saved search can be limited to the current page, or in some cases available in other searches for the same object.

Changing Settings or Renaming and Deleting Saved Searches
Follow these steps:
1. Click the Show Filters link if filters are currently hidden.
2. Select Manage from the Saved Search list.
3. Define settings for any saved search, predefined or user-defined, in the Manage Saved Searches dialog box:
   - Default: The saved search is automatically selected whenever you open this page.
• **Show in Saved Search List**: The saved search is available for you to select and run on this page. You can still find hidden saved searches in the Manage Saved Searches dialog box.

  | **Note**: Changes you make to predefined saved searches apply only to you, unless you’re editing the saved search within Page Composer.

4. The **Active** setting identifies the saved search that’s currently selected on the page. You can designate a different active saved search to have that saved search automatically selected as soon as you click **OK** in this dialog box.
5. For user-defined saved searches only, you can also rename or delete the saved search.
6. Click **OK**.

## Simplified Pages Layout and Content

### Modifying Simplified Pages Using Page Composer: Procedure

On a simplified page, you can modify user interface (UI) components by updating their properties, for example to change field labels, hide the component, or make a check box required.

When you start modifying simplified pages, by default, you can use the Design view. In the Design view, you can add content and make layout changes only in some pages. For other pages, you must use the Source view to make such changes.

  | **Note**: Any changes you make apply:
  | • Only to the page you’re on.
  | • To all or specific groups of users, depending on the context layer you select before making changes.

### Prerequisites

**Activate a sandbox.**

### Modifying a UI Component

To update component properties:

1. Click your user image or name in the global header and select **Customize Pages**.
2. Select a context layer, for example to make changes only for users with a specific job role.

  | **Note**: When you modify a UI component for a specific job role, that job role must be assigned to you for you to test the application change in the sandbox. Your security administrator can either assign the job role to you directly, or make the job role self-requestable for you to add it yourself from the resource directory.

3. By default, you start in the Design view, which lets you navigate to the component you want to modify.

   You can tell you’re in this view when the Design button on top of the page is highlighted. To use the Source view, you must select **Source** from the View menu. This menu isn’t displayed by default. To display the View menu, and then select Source, set the **Source View for Page Composer Enabled** profile option (FND_PAGE_COMPOSER_SOURCE_VIEW) to Yes.
4. When you have found your UI component, click the **Select** button on top of the page.
5. Hover over the UI component until a border appears around the component, and click.
6. Select **Edit Component**.
7. Update the component’s properties to make the application change you want.
Each component has its own set of properties, which may include some of the properties in this table. In Design view, you get the main properties only; but in Source view, you get all properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Text used by screen readers, for information in addition to what is provided in the Short Desc property.</td>
</tr>
<tr>
<td>Label</td>
<td>Display text for the component, for example the field prompt or the single prompt for a group of check boxes.</td>
</tr>
<tr>
<td>Read only</td>
<td>Whether users can edit the component, for example if a check box can be selected or not.</td>
</tr>
<tr>
<td>Rendered</td>
<td>Whether the component is visible or hidden to users on the page.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether users must enter something for the component before saving the page.</td>
</tr>
<tr>
<td>Short Desc</td>
<td>Text that appears when users hover or focus on the component, for example hover over a field label or click in the text box.</td>
</tr>
<tr>
<td>Show Component</td>
<td>Whether the component is visible or hidden to users.</td>
</tr>
<tr>
<td>Show Required</td>
<td>Whether an asterisk is displayed to indicate that the component is required.</td>
</tr>
</tbody>
</table>

8. To modify more components, click **Add Content** to return to the Design mode and repeat steps 4 to 7.

9. Click **Close** to save.

If available, click **Save and Label** instead to also label your changes so that you can later revert to the application changes you’re saving. Labels are saved with a prefix of `composer_`. For example, if you enter `myLabel`, then the label is `composer_myLabel`.

**Related Topics**

- Working With Context Layers: Examples
- Setting Up Sandboxes: Procedure
- Configuring Navigation and Home Page: Overview
- Role Provisioning and Deprovisioning: Explained

### Creating and Managing Announcements: Procedure

Use the Announcements page to create, edit, and delete announcements.

**Creating Announcements**

To create an announcement:

1. From the Navigator menu, select **Tools > Announcements**.
2. Click Create.
3. Enter a subject.
4. You can also specify the start and end date.
5. Select a category. If you select User-Defined, you can provide additional details in the text box.
6. To add an image, select:
   - **Predefined**: Select a file from the list of predefined images.
   - **File**: Browse and select a file from your local computer.
   - **URL**: Enter a full URL for the image
7. Add the content in the text box.

**Tip**: Once you have entered the content, you can use the available content formatting options.

8. Click Save and Close.

Your changes on the Announcements page apply immediately to all users, even if you saved your changes while a sandbox is active and not yet published.

**Editing and Deleting Announcements**

Use the Edit Announcement page to edit and delete announcements. To open this page, click an announcement on the Announcements page.

**Viewing Announcements on the Home Page**

You can determine if announcements are displayed on the home page, using the Home Page Layout page. To open this page from the Navigator menu, select **Configuration > Appearance**, and then click the Home Page Layout tab. To display announcements on the home page:

- Select **Announcements** from the home panel options, if your home page layout is panel or banner.
- Make **News and Announcements** visible, if your home page layout is News Feed.

Your default home page layout also determines how the announcements are displayed on the home page. If your default home page layout is:

- **Panel or Banner**: Only the announcement’s content (not subject or image) appears.
- **News feed**: The entire announcement along with the subject and image appears in the News and Announcements section.

**Related Topics**

- Why can’t I see announcements on the Home page?

**New Page Creation**

**Creating Pages for Hosting Third Party Applications: Procedure**

Use the Page Integration pages to create pages for hosting third party applications to address needs specific to your organization. All these new pages are grouped in a single group on the Navigator menu and springboard.
Prerequisites
Activate a sandbox.

Creating the First Page
Follow these steps:

1. From the Navigator menu, select Configuration > Page Integration.
2. Click New Page.
3. On the Create Page page, enter a page name.
4. In the Category Name field, enter a name for the group to place your page in.

**Note:** You can move your new pages to other groups later using the Structure page.

5. Search and select an icon for the page.
6. From the list of application roles, select the role to whom you want to grant access to the page. This list contains only custom roles because you can’t modify predefined roles.
7. In the Web Page field:
   - Enter the application URL that you want to host on this page.
   - Alternatively, you can create a secure web page URL:
     i. Click the Create Secure Web Page URL icon.
     ii. Select the name of the web application.
     iii. Enter the destination for the web application.
     iv. Enter a secure token name.
     v. Save and close the Create Secure Web Page URL dialog box.

In a secure web page, the application validates the secure token and uses it to authenticate web services within the end-user context. Using this mode of modified access, a partner can directly perform an action or display information to the specific user without any additional authentication.

8. Click Save and Close.

New pages are secure. Your security administrator must assign the privileges shown on the Page Integration Overview page to users such that they can access these pages.

If you have only one page in a group, then that page icon appears at the top level (not in any group) on the springboard. However, such page icons appear in their respective categories on the Navigator menu.

Creating Subsequent Pages
After creating the first page, follow these steps to create more pages:

2. Enter a page name.
3. Follow steps 5 to 7 in the procedure for creating the first page.
4. Click Save and Close.

After you have created the first page, all subsequent pages are added in the same group as that of the first page, by default.

Related Topics
- Setting Up Sandboxes: Procedure
Managing Pages Hosting Third Party Applications: Procedure

After creating pages for hosting third party applications, manage them using the options available on the Page Integration Wizard: New Pages page and the Page Integration Wizard: Existing Pages page.

You can do the following actions:

- Edit page settings.
- Add tabs to new and existing pages.
- Edit page tabs.
- Rename Categories.
- Navigate to pages.

Prerequisites

Activate a sandbox.

Editing Page Settings

Follow these steps:

1. On the Page Integration Wizard: New Pages page, click the name link for the page that you want to edit.
2. On the Edit Page page, make the required changes.
3. Click Save and Close.

You can make the following changes to a page:

- Change the icon for the page.
- Change the web page URL that you want to host on this page.
- Add tabs to the page.
- Delete the page.

Note: If a page has additional tabs, apart from the one created by default with the page, then you can delete the page only after deleting its tabs.

- Edit the page tabs.

Adding Tabs to New and Existing Pages

When you create a page, a page tab is created by default. You can then add more tabs to your new and existing pages, as required.

To add tabs to new pages, follow these steps:

1. On the Page Integration Wizard: New Pages page, click the name link for the page containing the tab that you want to edit.
2. On the Edit Page page, click Add Tab.
3. On the Create Tab page, enter a tab name.
4. Search and select an icon for the page.
5. From the list of application roles, select the role to whom you want to grant access to the page. This list contains only custom roles because you can’t modify predefined roles.
6. In the Web Page field:
   - Enter the application URL that you want to host on this page.
   - Alternatively, you can create a secure web page URL:
     i. Click the Create Secure Web Page URL icon.
     ii. Select the name of the web application.
     iii. Enter the destination for the web application.
     iv. Enter a secure token name.
     v. Save and close the Create Secure Web Page URL dialog box.

7. Click Save and Close.

To add tabs to existing pages, follow these steps:

1. Click the Page Integration Wizard: Existing Pages icon on the left region of the Page Integration Wizard: New Pages page.
2. Click Add Tab to Existing Page.
3. In the Select Page dialog box, select a page to add a new tab to.
4. Follow steps 2 to 7 in the procedure for adding tabs to new pages.

Editing Page Tabs

To edit page tabs, follow these steps:

1. On the Page Integration Wizard: New Pages page, click the name link for the page containing the tab that you want to edit.
2. On the Edit Page page, click the name link for the page tab that you want to edit.
3. On the Edit Tab page, make the required changes.
4. Click Save and Close.

You can make the following changes to a page tab:

- Change the icon for the page.
- Change the web page URL that you want to host on this page tab.
- Delete the page tab.

Renaming Categories

All pages that you create using the Page Integration Wizard: New Pages page are grouped in a single category. To rename the category for all pages, click Rename Category on the Page Integration Wizard: New Pages page.

Navigating to Pages

On the Page Integration Wizard: New Pages page, click the icon for the page that you want to navigate to, and view its content.

Related Topics

- Setting Up Sandboxes: Procedure
- Configuring Links for Page Entries: Procedure

Modifying User Interface Text
Selecting a Tool for Text Changes: Points to Consider

You can modify and replace words or phrases that appear on pages, in messages, and other areas of user interface using several tools or tasks.

Following are the tools for making text changes:

- Application Composer
- User Interface Text
- Page Composer

Multiple factors influence the option you select. For example:

- The offering you use
- The extent and scope of your changes
- The components that you modify

This table presents the navigation and offering availability options associated with the tools you can use to modify user interface text.

<table>
<thead>
<tr>
<th>Tool or Task</th>
<th>Navigation</th>
<th>Offering Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Composer</td>
<td>In the Navigator, select Configuration &gt; Application Composer.</td>
<td>Oracle Sales Cloud</td>
</tr>
<tr>
<td>User Interface Text</td>
<td>In the Navigator, select Configuration &gt; User Interface Text.</td>
<td>All applications</td>
</tr>
<tr>
<td>Page Composer</td>
<td>Click your user image or name in the global header, and select Customize &lt;Page Name&gt; Pages in the Settings and Actions menu. If the Customize &lt;Page Name&gt; Pages option isn’t available in the Settings and Actions menu, then select Customize Pages instead.</td>
<td>All applications</td>
</tr>
</tbody>
</table>

Text Modification Scenarios

The following table includes possible scenarios for modifying user interface text. Compare your situation to the scenario in the table to determine the most appropriate tool for modifying text in your application.

<table>
<thead>
<tr>
<th>Task</th>
<th>Scope</th>
<th>Tool or Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simultaneously replace multiple occurrences of a word or phrase that appear on multiple pages in multiple contexts</td>
<td>Comprehensive. The changes affects multiple pages throughout your application. You can edit the embedded help (for example, hints) using this method.</td>
<td>User Interface Text</td>
<td>Change the word “Employee” to “Associate” on every page associated with Employee Self Service, Benefits, and Payroll.</td>
</tr>
<tr>
<td>Task</td>
<td>Scope</td>
<td>Tool or Task</td>
<td>Example</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Simultaneously replace multiple occurrences of a word or phrase that appear in parts of messages in the message dictionary</td>
<td>Comprehensive.</td>
<td>User Interface Text</td>
<td>Change the word &quot;Employee&quot; to &quot;Associate&quot; in every message associated with Employee Self Service, Benefits, and Payroll.</td>
</tr>
<tr>
<td>Simultaneously replace multiple occurrences of the singular and plural forms of a word or phrase that appear in messages and on pages</td>
<td>Comprehensive.</td>
<td>User Interface Text</td>
<td>Change the word &quot;Employee&quot; to &quot;Associate&quot; and &quot;Employees&quot; to &quot;Associates&quot;.</td>
</tr>
<tr>
<td>Replace a word or phrase that appears on a specific page</td>
<td>Targeted: A page</td>
<td>Page Composer</td>
<td>Change the word &quot;Customer&quot; to &quot;Account&quot; on two specific pages.</td>
</tr>
<tr>
<td>Replace a word or phrase that appears in a specific message in the message dictionary</td>
<td>Targeted: A message</td>
<td>Manage Messages task</td>
<td>Change the word &quot;Recruit&quot; to &quot;Potential Employee&quot;, but only in two specific messages. All other messages continue to use the word &quot;Recruit&quot;.</td>
</tr>
<tr>
<td>Simultaneously replace a word or phrase associated with a specific object wherever the object appears</td>
<td>Targeted: A business object</td>
<td>User Interface Text</td>
<td>In Oracle Sales Cloud, change the label of the opportunity business object, from &quot;Opportunity&quot; to &quot;Deal&quot;. You want the change to affect the business object wherever it appears.</td>
</tr>
<tr>
<td>Replace words or phrases that appear in menus and menu items</td>
<td>Targeted: Navigator menu item text</td>
<td>User Interface Text</td>
<td>Change the menu item label from &quot;Total Compensation Statements&quot; to &quot;Compensation Package Statements&quot;.</td>
</tr>
</tbody>
</table>

Regardless of the tool you use to make changes, all application changes are written in a single override bundle. Hence, the latest application changes overwrites the previous ones.

**Note:** If you replace text using plain text as the input value, it supersedes any changes that use the override bundle. For example, if you enter a direct string in Page Composer, then Page Composer writes these changes in a file (not string resource) containing page modifications. Hence, such changes overwrite the changes in the override bundle.

**Related Topics**

- Tools for Configurations and Extensions: Critical Choices
Bulk Text Modifications: Explained

Use the User Interface Text tool to simultaneously update multiple occurrences of entire words or phrases in the user interface (UI).

You can use this tool to do the following activities for bulk text modification:

- Sandbox integration
- Case-sensitive and whole word searches
- Singular and plural text replacement
- Contextual previews

🎀 Note: You must activate a sandbox to use the User Interface Text tool.

To use the User Interface Text tool, on the Navigator, select Configuration > User Interface Text. Then, click Search and Replace to search and replace texts in bulk. The User Interface Text tool searches text on pages and in messages in the message dictionary. The search includes user assistance only if the user assistance text is in the message dictionary. The modification functionality for this tool doesn't extend to text in service oriented architecture (SOA) processes.

In the User Interface Text tool, you can:

- Search and replace
- Preview and adjust
- Save and publish

Search and Replace

After you activate a sandbox and click Search and Replace, enter the search text and the replacement text. You can enter the singular and plural forms of whole words or phrases. You can also use the following check boxes:

- Match Case - To perform case-sensitive searches.
- Match Complete Word or Phrase - To search for an exact match of your search text value.

🎀 Note: You can't perform partial word searches, nor can you use wildcard characters as part of the search text.

The following table lists the sample values that you can use as a guide while entering search text.

<table>
<thead>
<tr>
<th>Search Text</th>
<th>Expected Match</th>
<th>Match?</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flex</td>
<td>flex</td>
<td>Yes</td>
<td>The application searches for any occurrence of your search string without regard to its position in the strings it searches. Unless you select Match Case, all matches are considered exact.</td>
</tr>
<tr>
<td>Flex</td>
<td>Flex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flex</td>
<td>flexfields</td>
<td>No</td>
<td>The application treats your search text value as a whole</td>
</tr>
<tr>
<td>flexfields</td>
<td>Flexfields</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search Text</td>
<td>Expected Match</td>
<td>Match?</td>
<td>Reason</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A flexfield is a...</td>
<td>A flexfield is a...</td>
<td></td>
<td>The text flex isn’t the same as the text flexfields.</td>
</tr>
<tr>
<td>^Flex$</td>
<td>flexfields</td>
<td>No</td>
<td>Use ^string$ in the search field to say this string must match the complete field.</td>
</tr>
<tr>
<td></td>
<td>Flexfields</td>
<td></td>
<td>The application treats your search text value as the entire value of the strings it searches.</td>
</tr>
<tr>
<td></td>
<td>A flexfield is a...</td>
<td></td>
<td>Alternatively, select <strong>Match Complete Word or Phrase</strong> to search for an exact match of your search text value.</td>
</tr>
<tr>
<td></td>
<td>A Flexfield is a...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>^Flex$</td>
<td>flex</td>
<td>Yes</td>
<td>Use ^string$ in the search field to say this string must match the complete field.</td>
</tr>
<tr>
<td></td>
<td>Flex</td>
<td></td>
<td>The application treats your search text value as the entire value of the strings it searches.</td>
</tr>
<tr>
<td></td>
<td>A flexfield is a...</td>
<td></td>
<td>Unless you select <strong>Match Case</strong>, both matches are considered exact.</td>
</tr>
<tr>
<td></td>
<td>A Flexfield is a...</td>
<td></td>
<td>Use just ^string in the search field to say this phrase must appear at the beginning of a field.</td>
</tr>
<tr>
<td>flex credits</td>
<td>Flex Credits Configuration</td>
<td>Yes</td>
<td>The application searches for the exact spelling and sequence of words without regard to their position in the strings it searches.</td>
</tr>
<tr>
<td></td>
<td>Allow rollover of flex credits</td>
<td></td>
<td>Unless you select <strong>Match Case</strong>, all matches are considered exact.</td>
</tr>
<tr>
<td></td>
<td>Flex Credits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manage Flex Credits Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>credits configuration$</td>
<td>Flex Credits Configuration</td>
<td>Yes</td>
<td>Use string$ in the search field to say this phrase must appear at the end of a field.</td>
</tr>
<tr>
<td></td>
<td>Manage Flex Credits Configuration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Include one or more of the following match categories in your search:

- User Interface Text
- Oracle Transactional Business Intelligence Repository Definition Text
- Enterprise Scheduler Text
- Global Menu Label Text
- Multipart Validation Message

After you enter the search text and replacement text, click Preview Changes. The tool looks for exact whole word matches.

Preview and Adjust

The preview sorts the search results and presents the matches on tabs based on match categories. Data grids on each tab present the matches in rows. You can adjust each row independently. The grids on each tab are similar, but not identical.

⚠️ Caution: Some tab names on the Preview Text Changes page are followed by an asterisk (*) (for example, the Messages tab). For these tabs, once you save the preview text changes, these text changes are applied to the mainline metadata, that is, outside the sandbox. So you can’t undo your changes after you preview and save them even though you are currently in a sandbox. Also, on destroying the sandbox, these changes remain as is.

Each row on all tabs includes:

- A view of the existing text and the immediately surrounding text for context. You can’t edit the existing view.
- A preview of the replacement text and the immediately surrounding text for context. You can edit the preview.
- An option to exclude the row and the specific match you see in the row from the change.

Each row on the Messages tab includes an indicator. This indicator identifies when the search text appears in a message subcomponent, not necessarily in the message body displayed in the row. You can expand the row and view subordinate rows that display the message subcomponent containing the match and the preview, in context as previously described.

For each row in the preview, you can:

- Do nothing to accept the change as you see it.
- Select Exclude to eliminate the row from the batch update and maintain the existing text.
- Modify the replacement text to fine-tune the change for the specific match in the row. The row remains part of the batch update, even if the actual update differs from the other rows.

In the Global Menu Label Text tab, you can’t update a secure JSON Web Token (JWT).

If you have multiple languages in your application and you want to make similar text changes in those languages:

1. Change your language preference
2. Search and replace text
3. Validate your changes

Save and Publish
After you review and adjust the matches:

1. Save your changes.
2. Thoroughly test the runtime pages to make sure that every occurrence of the text is replaced, as you wanted.
3. Publish the sandbox.

Note the following points:

- Don’t publish a sandbox before you visually inspect and validate all pages and messages that contain text that you updated.
- Users can view:
  - Message, business intelligence, and enterprise scheduler text modifications when you save them, even if you don’t publish the sandbox.
  - Page text modifications when you publish the sandbox.

Related Topics
- Setting Up Sandboxes: Procedure

Adding Translations of Modified Text: Overview
If you install and use multiple languages in your application and you modify text, then enter translations of the modified text for all languages. You can either add translations at runtime, or export strings for offline translations. You can enter text translations for existing and newly added strings manually at runtime.

You can use several configuration tools for updating or adding strings. For example, you can use lookups to add translations at runtime. However, you can use the User Interface Text tool to update all strings and enter translations, as well as to perform offline translations.

Related Topics
- Translating Flexfield and Value Set Configurations: Explained

Translating Existing Strings at Runtime: Worked Example
This example demonstrates how to translate existing strings manually at runtime.
The following table summarizes the key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In this Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s the sandbox name that you want to use for translating existing strings?</td>
<td>Sandbox1</td>
</tr>
<tr>
<td>What’s your base language?</td>
<td>English</td>
</tr>
</tbody>
</table>
Decisions to Consider | In this Example
--- | ---
What’s the existing text that you want to modify? | Page
What’s the replacement text that you want to replace the existing text with? | Work area

Entering Text Translations for Existing Strings

1. Activate Sandbox1.
2. On the Navigator, select **Configuration > User Interface Text**.
3. Click **Search and Replace**.
4. In the Search For field, enter the text, "page".
5. In the Replace With field, enter the text, "work area".
6. Click **Preview Changes** to preview and adjust the matches, as necessary.

⚠️ **Caution**: Some tab names on the Preview Text Changes page are followed by an asterisk (*) (for example, the Messages tab). For these tabs, once you save the preview text changes, these text changes are applied to the mainline metadata, that is, outside the sandbox. So you can’t undo your changes after you preview and save them, even though you’re currently in a sandbox. Also, on destroying the sandbox, these changes remain as is.

7. Save your text changes.

✏️ **Note**: Repeat steps 4 to 7 for any text changes required in other installed languages.

8. Test and verify all messages and pages affected by the text changes. Be sure to test across all applications.
   Your replacement text for the existing string is now available to all users.

Translating New Strings Added Using Configuration Tools: Worked Example

This example demonstrates how to translate new strings that were added using configuration tools. While creating strings using configuration tools, such as the Structure page, always use the same language, that is, your base language.

The following table summarizes key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In this Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s the sandbox name that you want to use for translating newly added strings?</td>
<td>Sandbox2</td>
</tr>
<tr>
<td>What’s the language you want to translate your newly added English string to?</td>
<td>French</td>
</tr>
<tr>
<td>What’s the newly created English text that you want to translate in French?</td>
<td>Computer</td>
</tr>
</tbody>
</table>
Entering Text Translations for Newly Added Strings

1. Activate Sandbox2.
2. From the Navigator menu, select **Configuration > User Interface Text**.
3. Select French as the language you want to translate your new English string to.
4. In the Search For field, enter the newly created English string "computer". You must search in English because the French equivalent has not yet been created.
5. Enter the French string "ordinateur" as the replacement text.
6. Click **Preview Changes** to preview and adjust the matches, as necessary.

⚠️ **Caution:** Some tab names on the Preview Text Changes page are followed by an asterisk (*) (for example, the Messages tab). For these tabs, once you save the preview text changes, these text changes are applied to the mainline metadata, that is, outside the sandbox. So you can't undo your changes after you preview and save them even though you’re currently in a sandbox. Also, on destroying the sandbox, these changes remain as is.

7. Save your text changes.
8. Test and verify all messages and pages affected by the text changes. While testing, you must sign in with French as the language and use sandbox2.

✏️ **Note:** Repeat steps 3 to 8 for every active language.


Your replacement text for the newly added string is now available to all users.

FAQs for User Interface Text Changes

**Can I undo text changes that I made using the User Interface Text tool if I haven't published the sandbox?**

It depends on types of text changes in the sandbox. You can undo all text changes done in the user interface and global menu label by destroying the sandbox before publishing it. However, you can’t undo the text changes done in messages, analyses and reports, and scheduled processes.

**Can I get a report of all modified text if I want to analyze, troubleshoot, and diagnose the cause of unexpected action?**

No, but you can use Configuration Setup Manager to export all your configurations to a .zip file. You can find the text changes in files ending in "*.xlf.xml". These files list all text changes done in your application using browser-based tools, such as Application Composer, Page Composer, and User Interface Text. You can use the contents of these files for diagnosis and troubleshooting purposes. These files are read-only, so you can’t edit their contents.
Why didn’t text in my BI reports and SOA processes change when I used the User Interface Text tool to perform comprehensive text updates?

The bulk updates that you perform using the User Interface Text tool affect only the text that appears on application pages and in message dictionary messages.

Modifying Themes

Configuring Themes and Home Page Settings: Overview

Use the Appearance work area to configure the general look and feel of your application, and the default layout and display settings of the home page. To open this work area, on the Navigator, select Configuration > Appearance. The Appearance work area has 2 tabs: Themes and Home Page Layout. Use the Themes page to configure the default home page layout, and to change the branding logo, background colors, icon styles, and so on. You can apply an existing theme to your application pages, or create your own theme and apply it. Use the Home Page Layout page to configure the display settings of the home page.

Prerequisites

Before changing themes or configuring home page layout, follow these steps:

1. On the Navigator, select Configuration > Appearance.
2. Activate a sandbox. If you’re not in an active sandbox, click Edit on the Appearance work area. You’re prompted to activate a sandbox.

Note: If you’re already in an active sandbox, then the Edit button doesn’t appear on the Appearance work area.

After changing your theme, you can preview and test your changes, and then publish the sandbox to make your changes available to users.

Configuring Themes and Default Home Page Layout

Use the Themes page of the Appearance work area to:

• Set the default home page layout as Panel, Banner, or News Feed. The home page with the:
  o Panel or banner layout contains a springboard with icons that you can use to open work areas. Also, based on setup, the home page shows either company announcements or social networking conversations. This information appears in a panel for the panel layout, and in a banner for the banner layout.
  o News feed layout mainly contains the Apps section and a series of updates with important information.

• Create and edit saved themes, that is, themes that are not predefined. For example, you can change the following look and feel aspects of your application pages:
  o Logo
  o Background image
  o Panel image and style
• Size and style of the icons on the springboard
• Style of the cards, which appear on a page in a grid view. These cards display a summary of a single record, with attributes on the front side and optional back side. You can specify whether all cards should display with a dark-colored or a light-colored finish for users.
• Shape of buttons, menus, and tabs
• Colors for the background, global region, headings, page links, and buttons

💡 Tip: While making changes on the Themes page, you can click **Apply** any time to preview your changes.

### Configuring the Display Settings of the Home Page

Use the Home Page Layout page of the Appearance work area to configure the display settings of the home page.

- For panel and banner layouts, you can specify the default content to be displayed on the home page panel and banner, and the display photo on the main panel of the home page.
- For the news feed layout, you can rename the home page sections, show or hide them, and reorder them.

**Related Topics**

- [Setting Up Sandboxes: Procedure](#)

### Creating Themes: Procedure

Use the Themes tab of the Appearance work area to create themes.

**Prerequisites**

Activate a sandbox.

**Creating Themes**

Follow these steps:

1. On the Navigator, select **Configuration > Appearance**.
2. From the **Themes** list, select your base theme.
3. Set the default layout of the home page, and configure the various appearance settings for your application, as required. For example, select a branding logo, and specify color schemes.
4. On the Actions menu, select **Save As**.
5. Enter a theme name.
6. Optionally, deselect **Apply this theme** if you don’t want to apply the theme to the application immediately.
7. Click **OK**. If you have selected the Apply this theme check box, then your theme is saved and set as the current theme. If you haven’t selected the check box, your theme isn’t applied to the application. However, the theme is saved, and you can apply it to your application later.

**Related Topics**

- [Setting Up Sandboxes: Procedure](#)
Managing Themes: Procedure

Use the Themes page to manage your themes. To open the Themes page, on the Navigator, select **Configuration > Appearance**.

You can edit saved themes, apply themes to your application pages, and delete saved themes. You can’t edit or delete predefined themes.

**Prerequisites**

Activate a sandbox.

**Applying Themes**

From the **Themes** list, select a theme, and click **Apply**. If the selected theme is a predefined one, then save it as a new theme, and then edit and apply the theme, as required.

**Applying the Default Theme**

On the Actions menu, select **Apply Default**. The default theme is applied to your application.

**Editing Themes**

Follow these steps:

1. On the Navigator, select **Configuration > Appearance**.
2. From the **Themes** list, select your base theme.
3. Set the default layout of the home page, and configure the various appearance settings for your application, as required. For example, select a branding logo and specify color schemes.
4. Click **Apply**. If the base theme you have selected is a:
   - Predefined theme, then enter a theme name, and click **OK** to create another theme with your modifications. This new theme is then applied to your application.
   - Saved theme, then your theme changes are directly applied to your application.

**Deleting Themes**

From the **Themes** list, select a saved theme that you want to delete, and then on the Actions menu, select **Delete**.

**Changing the Logo and Background Image**

Use the Themes page to define the:

- **Branding logo**, which appears above all application pages. You can use a logo of any size, but the recommended width is lesser than 200 px and height is lesser than 50 px.
- **Watermark**, which appears in the background of all simplified pages. Use an image that's as close to 1024 by 768 pixels as possible.

To select a logo and a background image, use one of the options:

- **File**: Browse and select a file from your local computer.
- **Predefined**: Select a file from the list of predefined images.
- **URL**: Enter a full URL for the logo or the watermark.
Changing the Logo and Color Schemes of the Application: Worked Example

This example demonstrates how to change a logo and the color schemes of an application using the Themes tab of the Appearance work area. Users see the logo in the global header.

Note: Changes made to the logo using Page Composer overwrite the changes done using the Appearance work area.

The following table summarizes the key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s the name of the new theme?</td>
<td>MyCompany</td>
</tr>
<tr>
<td>Which existing theme are you going to base this theme on?</td>
<td>Default</td>
</tr>
<tr>
<td>Which default home page layout are you going to use?</td>
<td>Banner</td>
</tr>
<tr>
<td>Which image are you going to use as the new logo?</td>
<td>MyCompany.png</td>
</tr>
<tr>
<td>Tip: You can use a logo of any size, but the recommended width is lesser than 200 px and height is lesser than 50 px.</td>
<td></td>
</tr>
<tr>
<td>Which background image are you going to use?</td>
<td>WatermarkOracle.png</td>
</tr>
</tbody>
</table>

Prerequisites

Activate a sandbox.

Changing the Logo and Color Schemes

You can specify the look and feel aspects of various UI elements, such as navigation icons and buttons, using the Themes page. The fields available for you to change these appearance settings may vary based on what you select for some fields. For example, if you select News Feed as the default home layout, the Group Indicator Color field isn’t displayed for selection because this field isn’t applicable for the news feed home page layout.

Follow these steps:

1. On the Navigator, select Configuration > Appearance. On the Appearance work area, click the Themes tab.
2. From the Themes list, select Default.
3. Select the default home layout as Banner.
Note: When you select a Panel or Banner layout, the home page contains a springboard with icons that you can use to open work areas. Also, based on setup, the home page shows either company announcements or social networking conversations. This information appears in a panel for the panel layout, and in a banner for the banner layout. When you select the News feed layout, your home page mainly contains the Apps section and a series of updates with important information.

4. From the Logo list, select File as the type of location where your logo is stored. Browse and select MyCompany.png.

5. From the Background Image list, select File as the type of location where your background image is stored. Browse and select WatermarkOracle.png.

6. To specify the general appearance of the application, enter the values as shown in this table, or select the colors from the color palette.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Region Background Color</td>
<td>A9A9A9 (that is, Dark Gray)</td>
</tr>
<tr>
<td>This field specifies the background color of the global header, which is the uppermost region in the user interface.</td>
<td></td>
</tr>
<tr>
<td>Global Text and Icon Color</td>
<td>4169E1 (that is, Royal Blue)</td>
</tr>
<tr>
<td>This field specifies the color of the text that appear in general and with icons in the global header.</td>
<td></td>
</tr>
<tr>
<td>Enable color gradient</td>
<td></td>
</tr>
<tr>
<td>Select the check box, specify the gradient type as Vertical, and start and end colors of the gradient as follows:</td>
<td></td>
</tr>
<tr>
<td>◦ Start Color: 00BFFF (that is, Deep Sky Blue)</td>
<td></td>
</tr>
<tr>
<td>◦ End Color: 4169E1 (that is, Royal Blue)</td>
<td></td>
</tr>
<tr>
<td>Cover Image</td>
<td></td>
</tr>
<tr>
<td>Select File as the type of location where your cover image is stored. Browse and select MyCover.png.</td>
<td></td>
</tr>
<tr>
<td>For the home page with the:</td>
<td></td>
</tr>
<tr>
<td>◦ Panel layout, this image appears on the main panel. The recommended image size is 344x622 px.</td>
<td></td>
</tr>
<tr>
<td>◦ Banner layout, this image appears on the banner. The recommended image size is 2600x290 px.</td>
<td></td>
</tr>
<tr>
<td>◦ News feed layout, this image appears in the Things to Finish section. The recommended image size is 2600x290 px.</td>
<td></td>
</tr>
<tr>
<td>Panel Style</td>
<td>Light</td>
</tr>
<tr>
<td>This field specifies the style of the main panel on the home page.</td>
<td></td>
</tr>
</tbody>
</table>

7. To specify the appearance of the navigation icons, enter the values as shown in this table, or select the colors from the color palette.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon Type</td>
<td>Solid</td>
</tr>
</tbody>
</table>
8. To specify the appearance of the application pages, enter the values as shown in this table, or select the colors from the color palette.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Color</td>
<td>8B008B (that is, Dark Magenta)</td>
</tr>
<tr>
<td>Link Color</td>
<td>0000FF (that is, Blue)</td>
</tr>
<tr>
<td>Selection Color</td>
<td>FFB6C1 (that is, Light Pink)</td>
</tr>
<tr>
<td>Highlight Color</td>
<td>FFB6C1 (that is, Yellow)</td>
</tr>
<tr>
<td>Card Style</td>
<td>Light</td>
</tr>
</tbody>
</table>

9. To specify the appearance of the buttons, enter the values as shown in this table, or select the colors from the color palette.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>000000 (that is, Black)</td>
</tr>
</tbody>
</table>
### Field Value

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border</td>
<td>000000 (that is, Black)</td>
</tr>
<tr>
<td></td>
<td>This field specifies the color of the button borders.</td>
</tr>
<tr>
<td>Enable color gradient</td>
<td>Select the check box and specify the start and end colors of the gradient as follows:</td>
</tr>
<tr>
<td></td>
<td>◦ Start Color: 4169E1 (that is, Royal Blue)</td>
</tr>
<tr>
<td></td>
<td>◦ End Color: 00BFFF (that is, Deep sky Blue)</td>
</tr>
<tr>
<td>Corner Rounding</td>
<td>Small</td>
</tr>
<tr>
<td></td>
<td>This field specifies the extent to which the button corners are rounded. For example, None indicates that the button corners are not rounded and Small indicates that the button corners are slightly rounded.</td>
</tr>
</tbody>
</table>

**Tip:** While making changes on the Themes page, you can click **Apply** any time to preview your changes.

10. On the Actions menu, select **Save As**.
11. Enter the theme name as **MyCompany**.
12. Make sure that **Apply this theme** is selected.
13. Click **OK**.

**Related Topics**
- Setting Up Sandboxes: Procedure
- Exploring Work Areas: Overview

### Defining Home Page Display Settings: Procedure

Use the Home Page Layout tab of the Appearance work area to define the display settings of the home page. To open the Appearance work area, from the Navigator menu, select **Configuration > Appearance**. The home page display options available on the Home Page Layout page may vary based on the default home layout setting that you have configured in the General section of the Themes page.

**Prerequisites**

Activate a sandbox.

### Defining Display Settings for Home Page with Panel or Banner Layout

If you have selected Panel or Banner as the default home page layout on the Themes page, then follow these steps:

1. Select one of the following options to display on the home page panel or banner:
   - **Social**: Displays social networking content, such as the number of followers
   - **Announcements**: Displays employee announcements
   - **Cover image**: Displays the image for the main panel or banner, which you specify on the Themes page
2. Specify whether to display the photo in the main panel or banner of the home page from the social network profile or from HCM.
3. Click **Apply**.

### Defining Display Settings for Home Page with News Feed Layout

Use the Home Page Layout page to define the display settings for sections of the home page with the news feed layout. You can define these settings only if you have selected **News feed** as the default home page layout on the Themes page.

Follow these steps:

1. Click the section name to rename it.
2. Click the **Visible** field for a section to change its visibility setting. You can show or hide the section on the home page:
   - **Yes**: The section appears on the home page.
   - **No**: The section doesn’t appear on the home page.
   - **EL expression**: The evaluation of the EL expression decides whether the section will appear on the home page.
3. Use the **Move Up** and **Move Down** icons to adjust the relative positions of the sections on the home page.
4. Click **Apply**.

### Related Topics

- Configuring Home Page Navigation: Procedure
- Setting Up Sandboxes: Procedure

### FAQs for Modifying Themes

**What happens to my theme if changes that affect themes are made using Page Composer?**

Application changes made using Page Composer overwrite the changes made using the Appearance work area.

⚠️ **Caution**: Before using the Appearance work area to change the look of your application, you must undo any changes affecting the theme done using Page Composer. Otherwise, the changes that you make using the Appearance work area may not show up in your application as you wanted.

### Modifying Infolets

**Creating and Managing Infolets: Overview**

Use infolets to aggregate key information for a specific area, for example, social networking or your personal profile. On the home page, to open infolets and view important information at a glance, you can:

- Click an icon in the page control, if your home page layout is panel or banner.
• Click an infolet tab in the Analytics section, if your home page layout is news feed.

On the home page, if infolet pages aren’t available in the page control or analytics section, you can enable them using the Home Configuration page in the Structure work area.

Creating and Managing Infolets
You can open the infolets page in Page Composer mode and perform the following tasks:

• Create infolets.
• Add content to infolets, for example:
  ◦ You can add a task flow or a performance tile report.
  ◦ If an infolet contains a performance tile report, then you can add a link to a detailed report in the same infolet.

  ✨ Note: In the context of infolets, report can mean analysis.

• Edit infolets:
  ◦ Edit infolet content.
  ◦ Add, change, or remove link to detailed report.
• Delete infolets

Enabling Icons for Infolet Pages
You can use the Home Configuration page to enable icons for infolet pages. Based on your default home page layout, these icons appear in the page control or the Analytics section on the home page. To open the Home Configuration page, from the Navigator menu, select Configuration > Structure, and then click the Home Configuration tab.

Related Topics
• Personalizing Infolets: Procedure
• Configuring Home Page Navigation: Procedure

Creating Infolets

Watch: This video tutorial shows you how to create infolets. The content of this video is also covered in text topics.

Creating Infolets: Procedure
Use the infolet page to create infolets and set their views.

Prerequisites
You must open the infolet page in Page Composer.
Follow these steps:

1. Activate a sandbox.
2. To open an infolet page on the home page:
   - Click an icon in the page control, if your home page layout is panel or banner.
   - Click an infolet tab in the Analytics section, if your home page layout is news feed.
3. On the infolet page, click your user image or name in the global header, and select Customize Pages in the Settings and Actions menu.
4. Select a context layer. The infolet page opens in Page Composer.

Note: Once you complete making changes, click Close to leave Page Composer. After testing your changes, you must publish the sandbox to make your changes available to users.

To create an infolet:

1. On the infolet page, click the Infolet Repository icon, and select Create Infolet.
2. Enter a title for the infolet.
3. Set the infolet views as follows:
   - Specify the dimensions for the front view.
   - Enable or disable the back view.
   - Enable or disable the expanded view, and specify its dimensions. The dimensions of the front and the back views must be the same, but the expanded view must be bigger because it displays more details.
4. Click Save and Close.
5. Add content to the infolet views and link a detailed report to the infolet.
6. Preview the infolet’s front view, drill down to the detailed report, and then preview the back and expanded view.
7. After testing your changes, publish the sandbox to make the new infolet available to users.

Related Topics
- Setting Up Sandboxes: Procedure

Managing Infolets: Procedure

On the infolet page, use the options available on each infolet to manage it. You can add and edit infolet content, add and edit detailed report links to infolets, and delete infolets. While adding and editing infolet content, you can’t add a business intelligence dashboard to an infolet because a dashboard report is generally bigger than an infolet.

Prerequisites
You must open the infolet page in Page Composer.

Follow these steps:

1. Activate a sandbox.
2. To open an infolet page on the home page:
   - Click an icon in the page control, if your home page layout is panel or banner.
   - Click an infolet tab in the Analytics section, if your home page layout is news feed.
3. On the infolet page, click your user image or name in the global header, and select **Customize Pages** in the Settings and Actions menu.
4. Select a context layer. The infolet page opens in Page Composer.

**Note:** Once you complete making changes, click **Close** to leave Page Composer. After testing your changes, you must publish the sandbox to make your changes available to users.

**Adding Content to Infolets**

You can add content to the following infolet views:

- Front
- Back
- Expanded

To add content to the infolet's front view, follow these steps:

1. Click **Add Content** on the infolet. This button is available on an infolet view only if it doesn't have any content in that view.
2. Search and select a performance tile or a task flow, and click **Add**. You can browse the business intelligence (BI) catalog to find the analytics and reports that you want to add.
3. Close the Add Content dialog box.

To add content to the infolet's back view, follow these steps:

1. Click the **Back View** icon on the bottom right corner of the infolet.
2. Click **Add Content** on the infolet's back view.
3. Follow steps 2 and 3 in the procedure for adding content to the infolet's front view.

To add content to the infolet's expanded view, follow these steps:

1. Click the **Expanded View** icon on the bottom right corner of the infolet's back view.
2. Click **Add Content** on the infolet's expanded view.
3. Follow steps 2 and 3 in the procedure for adding content to the infolet's front view.

**Editing Infolet Content**

You can edit the tile content of an infolet's:

- Front view
- Back view
- Expanded view

Follow these steps:

1. Click the **Actions** icon on the top right corner of the infolet's front view, back view, or expanded view, and select **Edit Content**.
2. Click **Add Content** to replace the existing content of the infolet.
3. Search and select a performance tile or a task flow, and click **Add**. You can browse the business intelligence (BI) catalog to find the analytics and reports that you want to add.
4. Close the Add Content dialog box.

**Linking Detailed Reports to Infolets**

To provide detailed information about a subject matter on an infolet, add a link to a detailed report. After you add the link, users can click anywhere in the infolet area to drill down to that detailed report. The detailed report doesn't replace the existing infolet content.
Adding Links to Detailed Reports

First, add a performance tile report to the infolet content, and then follow these steps:

1. Click the Actions icon on the top right corner of the infolet, and select Link Detailed Report.
3. Search and select a report, and click Add to add it to the infolet.
4. Close the Add Content dialog box.
5. Click Done.

Editing Detailed Report Links

To edit a detailed report link in an infolet:

1. Click the Actions icon on the top right corner of the infolet, and select Edit Detailed Report.

Tip: To remove the report link, click Remove Report.
3. Click Done.

Deleting Infolets

To delete an infolet, click the Actions icon on the top right corner of the infolet, and select Delete.

Editing Infolet Visibility

You can show or hide an infolet on the infolet page. To edit the visibility settings of an infolet:

1. Click the Actions icon on the top right corner of the infolet, and select Edit Visibility.
2. Select one of the following:
   - Yes: The infolet appears on the infolet page.
   - No: The infolet doesn’t appear on the infolet page.
   - EL expression: The evaluation of the EL expression decides whether the infolet will appear on the infolet page.

Related Topics
- Setting Up Sandboxes: Procedure
- Personalizing Infolets: Procedure

FAQs for Modifying Infolets

Why is the icon for my infolet page not available in the page control on the home page?
The icon for your infolet page may be hidden. Change the visibility setting of the icon using the Home Configuration page of the Structure work area. To open this page, select Configuration > Structure from the Navigator menu, and then click the Home Configuration tab.

Related Topics
- Configuring Home Page Navigation: Procedure
How can I rename an icon for an infolet page in the page control on the home page?

You can rename an icon for an infolet page using the Home Configuration page of the Structure work area. To open this page, select Configuration > Structure from the Navigator menu, and then click the Home Configuration tab.

Related Topics

- Configuring Home Page Navigation: Procedure

What’s the difference between a performance tile report and a detailed report added to the infolets content?

Performance tile report shows data in the small infolet format. When you add a performance tile report to an infolet, users can see only the summary information about the subject matter. But this report doesn’t provide detailed information.

To provide detailed information about the subject matter on the same infolet, add a link to a detailed report. Users can click this link to gather more information.
10 Using Page Composer in Oracle Sales Cloud

Overview

Configure the look and feel of user interface (UI) pages using Page Composer. Create changes on a single page that all users can see, or only a subset. Create saved searches, or saved lists, to control which records display in summary tables. This chapter covers:

- Points to consider before you configure pages
- How to configure standard desktop pages, landing pages, and dashboards
- How to configure simplified pages
- How to use Direct Selection mode to make user interface changes
- How to work with components for configure pages

Note: Page Composer supports two editing modes: Design View and Source View. In Oracle Sales Cloud, you can make configuration changes only in the Design View mode. The only exception is the Partner Relationship Management’s Partner Portal UI Shell configuration, which is done in Source View mode.

Related Topics

- Modifying Simplified Pages Using Page Composer: Procedure

Configuring Oracle Sales Cloud UIs: Points to Consider

You can use multiple tools to configure Oracle Sales Cloud user interfaces. This topic highlights the differences between the types of configurations you can perform in two of these tools: Oracle Application Composer and Oracle Page Composer (also referred to as Composer in some documentation).

Main Page Composer and Application Composer Differences

This table outlines key differences between configurations you can make using Page Composer and Application Composer.

<table>
<thead>
<tr>
<th>Key Differences</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring the page you are viewing</td>
<td>Configure a UI page you are viewing and see your changes reflected immediately on that page.</td>
<td>You must work in a separate work area on application objects and you must test your work by navigating to the appropriate page.</td>
</tr>
<tr>
<td></td>
<td>Page Composer allows you to make changes strictly page by page. For example, if you make a field required on the Edit Opportunity page, the field is not automatically required on the Create Opportunity page.</td>
<td>Because Application Composer configures application objects, making a field required affects all interfaces where that field appears.</td>
</tr>
</tbody>
</table>
### Key Differences

<table>
<thead>
<tr>
<th>Configuring dashboards</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the format and content of dashboard pages. Dashboards can display BI Analytics and BI Publisher reports, notifications, as well as external content from your intranet or from the Internet.</td>
<td></td>
<td>Cannot configure dashboard pages.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported interfaces</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Simplified UI • Desktop UI</td>
<td>• Simplified UI • Desktop UI • Mobile • Microsoft Outlook</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Creating your own fields and application objects</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not supported.</td>
<td>You can add custom fields and entirely new objects.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Creating saved lists</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and edit saved lists (also called saved searches) for users. The saved lists provide saved search criteria to speed up common searches.</td>
<td>Not supported.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tailoring your configurations for users with different job roles</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides different levels of configurations for both simplified UI and desktop UI.</td>
<td>Supports configurations by job role or for all users.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuring external-facing partner pages</th>
<th>Page Composer</th>
<th>Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>You must use Page Composer to configure all external-facing partner pages, such as the partner registration page and the partner portal landing page.</td>
<td>Only for partner pages internal to your organization, such as the partner Overview page.</td>
<td></td>
</tr>
</tbody>
</table>

### What You Cannot Do in Page Composer

Using Page Composer, you cannot:

- Change either the data or functional security for a page. While you can hide some components, you must change the security setup to control access.
- Filter or reorder lists of values using Page Composer
- Add any code, validation logic, or scripting to the buttons or fields
- Add calls out to other APIs to populate fields with values
- Change the selections which appear in the Navigator menu. Instead, use the Review Menu Configurations task from the Setup and Maintenance work area.
- Configure the Search region inside Search and Select dialog.

### Can I use Page Composer to configure all elements on a page?

No, you cannot use Page Composer to configure all UI elements on a page to protect the underlying application logic.
You cannot select some page components in Page Composer’s Select mode. For others, the Edit Component menu is disabled or only a subset of the properties are available in the Component Properties window.

⚠️ **Caution:** Do not edit the **Taskflow Id** parameter in Page Composer Task List properties. Doing so may damage page metadata.

In general, you can configure at least some of the properties of the following page components:

- af:activeOutputText
- af:inputComboboxListOfValues
- af:inputDate
- af:inputFile
- af:inputListOfValues
- af:inputNumberSlider
- af:inputNumberSpinbox
- af:inputRangeSlider
- af:goLink
- af:query
- af:inputText
- af:outputFormatted
- af:outputText
- af:panelLabelAndMessage
- af:panelFormLayout
- af:panelHeader
- af:richTextEditor
- af:selectBooleanCheckbox
- af:selectBooleanRadio
- af:selectManyCheckbox
- af:selectManyChoice
- af:selectManyListbox
- af:selectManyShuttle
- af:selectOneChoice
- af:selectOneListbox
- af:selectOneRadio
- af:selectOrderShuttle
- af:showDetailHeader
- af:table
- af:column
- fnd:applicationsTable
- af:commandMenuItem
- af:menu
- af:panelGroupLayout
Configuring Oracle Sales Cloud Pages: Explained

You can use Oracle Page Composer to configure pages in a sandbox while you are working in the application. You can make your UI configurations available to all users or to a subset of users, for example, only to partners or to users with a specific job role.

Oracle Page Composer is a page editor, developed as part of Oracle WebCenter, for revising the layout and content of application pages. Its full capabilities are described in the Oracle Fusion Middleware User’s Guide for Oracle WebCenter and in the Oracle Fusion Applications Extensibility Guide.

In Oracle Sales Cloud, you can use Page Composer, also known as Oracle Composer, for performing the following types of configurations:

- Simple UI editing functions, such as showing and hiding fields, changing field labels, and making fields read-only or required
- Changing page layout and adding Oracle BI Publisher reports and other content to dashboard pages
- Creating saved searches (also called saved lists)
- Configuring the display of search results in tables

**Note:** If you are planning to configure the UI for a specific job role, then you must have that job role assigned to you so that you can test the configuration in the sandbox.

Opening Page Composer

To open page composer, do the following:

1. Activate a sandbox, as follows:
   a. Click your user image or name.
   b. From the **Settings and Actions** menu, select **Manage Sandboxes** under the **Administration** heading.
   c. If no sandbox is available, then create a sandbox by clicking **New** (plus sign) on the toolbar and entering a sandbox name.
   d. Select the sandbox name and click **Set as Active**.
      
   The active sandbox name appears at the top of the page.
2. Click the user image or name again and select **Customize Pages** from the **Settings and Actions** menu.

Page Composer Modes

In Oracle Sales Cloud, you can use Page Composer in one of two modes: the Add Content mode and the Select mode. The following figure shows a partial screen capture of the Page Composer toolbar with the two modes highlighted by callouts.

- Use the Add Content mode (highlighted by callout 1 in the following figure) to move, add, and edit regions on dashboards and external-facing partner pagers. You also use this mode to create saved searches, specify search result content, and to navigate in the UI when you are in Page Composer.
You are in the Add Content mode when you first open Page Composer. You can also select this mode by clicking **Add Content** on the top left corner of a page.

- Use the Select mode (callout 2), which you can activate by clicking **Select**, to configure the properties of different UI elements such as fields and table columns.

---

**Working in the Add Content Mode**

Use the Add Content mode to do the following:

- Configure the content on dashboard pages, including the Welcome page in the desktop UI.
  
  Dashboards pages are the pages that users land on when they navigate to work areas using the Navigator. Dashboard pages display analytics, notifications and other information.

- Configure all of the external-facing landing and transactional partner management pages.

- Configure the table display of search results.

- Create saved searches (saved lists).

- Navigate to other pages while working in Page Composer.

When you open Page Composer, the application opens in the Add Content mode automatically. On pages that support configuration in this mode, such as the sales dashboard in the following image, different regions are highlighted with borders and display configuration controls.

The following figure shows a partial screen capture of an application page with the different Add Content features highlighted by callouts.

- A bar appears at the top of the page indicating that you are in editing the page in Page Composer (callout 1).

- You can change the column layout of the page by clicking **Change Layout** (callout 2).

- You can move a region, by grabbing its toolbar (callout 3) and dragging it to a different position.

- You can add a new tab (callout 4).
You can also make changes in each region:

- You can remove a region by clicking Close, the icon at the top right-hand corner of the region.
- You can add content by clicking **Add Content**.
- You can add a new region by clicking one of the icons to the right of the **Add Content** button. These icons specify where in the page the new region is created.
The following figure shows a screen capture of a region on an application page when you are adding a region. The icons indicating where the region is created are highlighted.

Working in the Select Mode

In the Select mode, a border appears around individual fields and other UI elements as you move your cursor over them.

Clicking within a border for a UI element that supports configuration displays a menu with one or both of the following options: Edit Component and Edit Parent Component.
The following figure shows a screen capture of a partial application page displaying the Edit Component menu in the Page Composer Select mode.

Not all UI elements support configuration, so these editing options are not always enabled.

You use the Component Properties window to make your configurations. Not all the configurations are available on all the UI elements, so the content of this window varies. The following figure shows a screen capture of the window with callouts highlighting different components.

- To make a field read-only, you select the **Read Only** option (callout 1).
- To make a field required, you select the **Required** option (callout 2).

**Note:** Before making a field required, the field must display a value. If it does not, then cancel out of this window and enter the value first.

- To hide the field, you deselect the **Show Component** option (callout 3).
• To change the field label:
  
  a. Click the **Label** field to open a menu.
  
  b. Select **Select Text Resource** from the menu.
The following figure shows a screen capture of a portion of an application page showing the Label field and the menu.

![Screen capture showing Label field and menu](image)

The Select Text Resource window, shown in the following figure, appears.

![Select Text Resource window](image)
Editing the Partner Portal UI Shell Using the Page Composer Source View

Channel partner portal administrators can use Page Composer’s Source View to configure the header and footer areas of the pages partners use to register themselves and sign in after they are registered. This is the only page in the Oracle Sales Cloud which supports this mode. You can use this mode to add a company logo, copyright information or links, for example.

To configure the partner portal UI shell, you must activate a sandbox and select the Update Partner Portal UI Shell task in the Channel dashboard.

To learn more about working in the Source view, see the Oracle Fusion Middleware User’s Guide for Oracle WebCenter.

c. Enter a name in the Key field. You can use this name to search for this label when you configure other pages.
d. Enter the new field label in the Display Value field.
e. Click OK.

Note: Some UI components are protected from updates to preserve the product business logic. For example, if a field is read-only because of a security privilege defined in the application, you cannot make it editable in Page Composer.
Making Configurations Available to Different User Groups

You can make your configurations available to different subsets of users.

When you launch Page Composer, you must select one of the following configuration layers:

- **Site**: Your configurations are visible to all users.
- **External or Internal**: Depending on your selection, your configurations are visible to either internal users (your employees) or to external users (your partners or anonymous users).
- **Job Role**: Your configurations are visible to users with the job role you select. The list of available job roles includes all the job roles assigned to you directly and any job roles that are inherited by those job roles.

*Note*: If you created your own job roles, then you must run the Retrieve Latest LDAP Changes process from the Scheduled Processes work area before the job roles are available for selection.

By default, configurations you made at a higher level are propagated to lower-level layers, unless you deselect these layers in the Include column.

*Note*: The Welcome page in the desktop UI does not support different configuration layers. You must configure this page at the site level.

**Related Topics**

- What’s Required for Testing Configurations in the Sandbox
- Configuring Which Columns Display in Desktop UI Search Results: Worked Example

Modifying Simplified Pages: Procedure

On a simplified page, you can modify user interface (UI) components by updating their properties, for example to change field labels, hide the component, or make a check box required.

When you start modifying simplified pages, by default, you can use the Design view. In the Design view, you can add content and make layout changes only in some pages. For other pages, you must use the Source view to make such changes.

*Note*: Any changes you make apply:

- Only to the page you’re on.
- To all or specific groups of users, depending on the context layer you select before making changes.

**Prerequisites**

Activate a sandbox.

**Modifying a UI Component**

To update component properties:

1. Click your user image or name in the global header and select Customize Pages.
2. Select a context layer, for example to make changes only for users with a specific job role.

*Note:* When you modify a UI component for a specific job role, that job role must be assigned to you for you to test the application change in the sandbox. Your security administrator can either assign the job role to you directly, or make the job role self-requestable for you to add it yourself from the resource directory.

3. By default, you start in the Design view, which lets you navigate to the component you want to modify.

You can tell you're in this view when the Design button on top of the page is highlighted. To use the Source view, you must select Source from the View menu. This menu isn't displayed by default. To display the View menu, and then select Source, set the Source View for Page Composer Enabled profile option (FND_PAGE_COMPOSER_SOURCE_VIEW) to Yes.

4. When you have found your UI component, click the Select button on top of the page.

5. Hover over the UI component until a border appears around the component, and click.


7. Update the component's properties to make the application change you want.

Each component has its own set of properties, which may include some of the properties in this table. In Design view, you get the main properties only; but in Source view, you get all properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Text used by screen readers, for information in addition to what is provided in the Short Desc property.</td>
</tr>
<tr>
<td>Label</td>
<td>Display text for the component, for example the field prompt or the single prompt for a group of check boxes.</td>
</tr>
<tr>
<td>Read only</td>
<td>Whether users can edit the component, for example if a check box can be selected or not.</td>
</tr>
<tr>
<td>Rendered</td>
<td>Whether the component is visible or hidden to users on the page.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether users must enter something for the component before saving the page.</td>
</tr>
<tr>
<td>Short Desc</td>
<td>Text that appears when users hover or focus on the component, for example hover over a field label or click in the text box.</td>
</tr>
<tr>
<td>Show Component</td>
<td>Whether the component is visible or hidden to users.</td>
</tr>
<tr>
<td>Show Required</td>
<td>Whether an asterisk is displayed to indicate that the component is required.</td>
</tr>
</tbody>
</table>

8. To modify more components, click Add Content to return to the Design mode and repeat steps 4 to 7.

9. Click Close to save.

If available, click Save and Label instead to also label your changes so that you can later revert to the application changes you're saving. Labels are saved with a prefix of composer_. For example, if you enter myLabel, then the label is composer_myLabel.
Changing Page Layout Using Page Composer: Procedure

Page layout defines the number, placement, and orientation of content regions on a page. Although you set the layout style while creating a page, for some layouts, you can change the layout style even after adding content to the page.

> **Note:** You can't change the page layout for all pages.

Prerequisites

Activate a sandbox.

Changing Page Layout

To change the layout of a page:

1. Open the page in Page Composer.
2. From the View menu, select **Design**.
3. Click **Change Layout**.
4. Select the new layout.

**Related Topics**

- Setting Up Sandboxes: Procedure

Configuring the Global Page Template: Explained

The global page template provides a common header area and the footer panel for all pages in your application. You can use Page Composer to configure the global page template. To open the global page template in Page Composer, click your user image or name in the global header, and select **Customize Global Page Template** from the Administration menu.

You can make the following changes to the global page template:

- Add components
- Edit components
  
  Example: Add expression language to hide the tags link
• Delete components
Example: Remove the tags link

Tip: When you move your cursor over the global page template, the areas that you can edit display a blue outline.

You have two separate global page templates - one for the desktop user interface (UI) and the other for the simplified UI. If you click Customize Global Page Template from a desktop page, you will configure the template for the desktop UI. Similarly, if you click Customize Global Page Template from a simplified page, you will configure the template for the simplified UI. Hence, to ensure a consistent look and feel for all pages, you must configure the global page template twice, once for each UI.

Adding Components to the Global Page Template
To add components to the global page template:

1. Open the global page template in Page Composer.
2. Select the portion of the global header to which you want to add a component, and click Add Content.
3. In the component catalog, select Components to display the list of available components.
4. Click the Add button associated with the component you want to add.
   The component appears in the global header.
5. Change component properties, as appropriate. For example, if you added the Text component, enter the text that you want to display.
6. After completing your changes, click Close. When prompted, click Save to save your changes.

Editing Components in the Global Page Template
To edit components in the global page template:

1. Open the global page template in Page Composer.
2. Select the component that you want to edit.
3. Click Edit.
4. Edit the component properties, and click OK to save your changes.
5. After completing your changes, click Close. When prompted, click Save to save your changes.

Deleting Components from the Global Page Template
To delete components from the global page template:

1. Open the global page template in Page Composer.
2. Select the component that you want to delete.
   When you move your cursor over the global page template, the areas that you can edit display a blue outline.
3. Click Delete. When prompted, click Delete to delete the component.
4. After completing your changes, click Close. When prompted, click Save to save your changes.

Editing Footers in the Global Page Template
To edit footers in the global page template:
Note: For simplified pages, you can’t modify footers in the global page template.

1. Open the global page template in Page Composer.
2. Select the left region of the footer area. In the Source region, make sure the spacer: 40px node is selected after you select the left region of the footer area.
3. In the Source region, select the panelGroupLayout: horizontal node that appears above the spacer, and click Add Content.
4. In the component catalog, select Components to display the list of available components.
5. Click the Add button associated with the component you want to add. The component appears in the footer.
6. Change component properties, as appropriate.
7. After completing your changes, click Close. When prompted, click Save to save your changes.

For information about making skin modifications, such as selecting a different color palette, see the Oracle Fusion Applications Extensibility Guide for Developers.

What do I do if Page Composer quits while I am navigating in the UI?

Navigating between different objects in the application with Page Composer open in the Design mode, can sometimes cause Page Composer to quit. When you no longer see the Page Composer bar at the top of your screen, simply restart Page Composer again. Your previous work is preserved in the active sandbox, so there is no loss of data. Page Composer sometimes quits because it was designed to configure one object at a time.
Providing Analytics for Sales Users: Overview

Oracle Sales Cloud provides analytics that help sales teams continuously monitor and interpret key performance indicators for their organizations. Prebuilt analytics provide metrics on areas such as the pipeline, opportunities, performance, forecasts, activities, and customers. All of the prebuilt analytics can be configured in Oracle Business Intelligence (BI), where they are stored. In addition, administrators can build their own analytics using the reports and analyses wizards. And for sales teams on the go, Oracle Mobilytics provides quick access to analytics on the Apple iPad, while Oracle Sales Cloud Mobile gives analytics on mobile devices and tablets.

Analytics in Sales Team Work Areas

In Sales Cloud, administrators can add analytics to sales user work areas in a variety of different ways. Signed-in sales users see the analytics the administrators have set up for them, and can search and add analytics as favorites on their own Analytics pages.

The following table shows the work areas to which analytics can be added:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales pages</td>
<td>These pages include five blank pages that administrators configure, and to which they add prebuilt or user-defined analytics.</td>
<td>Not visible until administrators enable the Sales Pages in Set System Options</td>
</tr>
<tr>
<td>Sales infolet pages</td>
<td>Infolet pages come prebuilt with role-based analytics and infolets. One page is supplied for each role.</td>
<td>Not visible until administrators enable the Sales Infolets in Set System Options</td>
</tr>
<tr>
<td>Object pages</td>
<td>Analytics can be added to the following business object work area pages:</td>
<td>To add analytics tabs to these pages, administrators must set variables on the analysis, enable the subtab or side tab for the page, and then add an analysis on the page’s analytics tab.</td>
</tr>
<tr>
<td></td>
<td>• Leads</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Opportunities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Forecasts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accounts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Households</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Contacts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Activities</td>
<td></td>
</tr>
<tr>
<td>Analytics pages</td>
<td>In their Analytics pages, users can add analyses themselves by searching for an analysis and making it a favorite.</td>
<td>Administrators make company-defined analytics available for users on the Analytics page by adding session variables on an analysis. These variables can show data specific to the signed-in user viewing her Analytics page.</td>
</tr>
</tbody>
</table>
Briefing Books

A briefing book is a collection of BI analyses or dashboard pages (which can contain reports) that you create and save for later access. The static snapshots give you a picture of what’s going on at the time that the analysis or dashboard page is added to the briefing book. You can download briefing books as PDF or MHTML for viewing or printing, and share them with others. The PDF file includes a table of contents for the book. Like analytics and reports, briefing books are stored in the BI catalog.

Scheduled Analytics

You can submit reports as scheduled processes, and you can set up agents to email analytics, briefing books, and dashboards.

Sales Cloud Mobile

Using Oracle Sales Cloud Mobile, sales personnel can access analytics and reports from the Home page of a mobile device. Analytics also are embedded contextually for accounts. The contextual reports include data on sales account revenue trends, sales account win/loss trends, and sales account win/loss reasons.

Mobilytics

Oracle Mobilytics provides sales managers with sales analytics and reports using interactive graphics on your mobile devices. The following are some of the things you can do on the go, on mobile with the mobile analytic tools.

- Perform “what if” analyses by moving opportunities between quarters, and achieve your quota with Forecast Shaper.
- Analyze sales stages to move large or important deals along and improve conversion rates with Pipeline Analyzer.
- Track the deals by team’s activities and focus on the right deals with Deals Radar.
- Manage the team’s performance by tracking their quota versus attainment with Team Tracker.
- Keep the forecast on track by analyzing pipeline and honing stale deals with Aging Monitor.
The following figure shows key analytics on mobile devices.

Additional BI Resources

To find more information about Oracle Business Intelligence, consult the online help and refer to the Oracle Sales Cloud Creating and Administering Analytics guide.

Related Topics

- Listing of Sales Cloud prebuilt analytics in downloadable format
- Creating and Administering Analytics for Sales

Building Your Own Analytics: Explained

Oracle Business Intelligence (BI) holds all the analytics that are added to work areas. There are tools to build your own analytics, as well as edit the prebuilt analytics. All of the analytics are built using subject areas. Subject areas are built around sets of key business questions for a particular context, such as pipeline, performance, quota, activity, and so on.

To get to BI:

1. Click Navigator then Reports and Analytics.
2. In the Reports and Analytics pane, select the Browse Catalog open book icon.
To begin creating analyses, in BI, you select **New** and then **Analysis**, and then select from a variety of subject areas that hold the data that you use for building your analytics.

This figure shows an example of the subject areas in BI.

Once you select the subject area, the subject area opens up and you can add the columns to the editor.

This figure shows the palette for creating new analytics, and the Sales - CRM Pipeline subject area with the related data objects to build your analytic.
Creating Your Own Subject Areas: Overview

Custom subject areas are built from the objects you create and manage in Application Composer. Administrators can use prebuilt subject areas to build analytics, or they can create their own subject areas. A subject area is a set of entities, attributes, and measures grouped in a particular context. For example, a context could be the context of "pipeline", and there is a subject area which holds all of the detail relative to a pipeline. There are quota subject areas, contact, and customer subject areas, and many more to choose from. Still, there are times when you need to build a particular subject area for a particular context unique to your business. In Sales Cloud you will find easy tools to access all the pieces you need to build your own subject areas.

Objects in Oracle Sales Cloud can be classified under broad categories of custom objects and standard objects. Every custom subject area has a primary object, which is the focus of any analytics that you later create. You can also optionally add child objects and related objects to the custom subject area, to meet your reporting needs. Custom subject areas support both custom and standard objects. The objects that you create are called custom objects and those delivered with prebuilt with your application are standard objects.

Creating your own subject areas is detailed in the Creating and Administering Analytics guide.

Related Topics

- Creating and Administering Analytics for Sales
• Custom Subject Areas: Explained
12 Configuration of Navigation

Configuring Navigation and Home Page: Overview

You can configure the Navigator and springboard, as well as define settings for the home page and springboard using the Structure work area. To open this work area, from the Navigator menu, select **Configuration > Structure**. This work area has 2 tabs: Navigation Configuration and Home Configuration. Use the Navigation Configuration page to configure the Navigator and springboard. Use the Home Configuration page to configure the home page.

Some of the key configuration tasks that you can do using the Structure work area are:

- Create, edit, and rename the Navigator menu items and springboard icons, which appear on the home page.
- Create a duplicate version of an existing predefined page entry and edit the duplicate page entry to meet your specific requirements.
- Change the visibility settings of the Navigator menu items and springboard icons, and reorder them.
- Change the visibility settings of the icons for infolet pages in the page control, which appear on the home page.
- Rename and reorder the icons for infolet pages in the page control.

Groups and Page Entries

To address needs specific to your organization, you can create or edit groups and page entries for the Navigator and springboard. For example, you may want to link page entries to web pages or external applications.

Groups and page entries are available on the springboard and navigator menu. A page entry is the navigator link or springboard icon that opens a page. A page is a single screen to perform related tasks. A few page entries may be categorized in a group. Depending on the number of page entries that you have access to, the page entries can appear at the top level (not in any group folder) on the springboard. If you have only one page entry in a group, then that page entry icon appears at the top level on the springboard. However, such page entry icons appear in their respective groups on the Navigator menu.

While creating or editing a page entry or group, you can use the **Springboard** field to specify whether to display them on the springboard. So, not all page entries and groups may appear on the springboard. The Navigator menu may have more page entries and groups than the springboard. If a page entry appears on both the Navigator menu and springboard, then you can use either of them to open the page. For page entries that don’t appear on the springboard, use the Navigator menu to open those pages.

Configuring the Navigator and Springboard

Use the Navigation Configuration page to configure the Navigator and springboard. You can do the following tasks on groups and page entries:

- Create
- Edit
- Show or hide
• Duplicate predefined page entries
• Reorder

\* Note: If a page has both desktop and simplified versions, by default, users open the simplified version of the page from the Navigator menu or springboard. However, to open the desktop page instead of the simplified page, set the Desktop Pages Version Enabled profile option (FND_CLASSIC_INTERFACE) to Yes.

Configuring the Home Page

Use the Home Configuration page to configure the icons for infolet pages in the page control on the home page. You can rename these icons, change their visibility settings, and reorder them.

Related Topics

• Setting Up Sandboxes: Procedure
• Signing In and Getting Started: Explained
• Setting Profile Option Values: Procedure

Navigator and Springboard Configuration

Configuring the Navigator and Springboard

\* Watch: This video tutorial gives you an overview of the Navigator and springboard and shows you how to configure their groups and page entries. The content of this video is also covered in text topics.

Creating Groups and Page Entries for Navigation: Procedure

Use the Navigation Configuration page to create groups and page entries for configuring the Navigator and springboard. You can do either of the following:

• Create a group and then create a page entry in that group.
• Create a page entry in an existing group or at the top level (not in any group).

Prerequisites

Following are the prerequisites:

1. From the Navigator menu, select Configuration > Structure.
2. Activate a sandbox. If you’re not in an active sandbox, click Edit in the Structure work area. You’re prompted to activate a sandbox.
Tip: If you’re already in an active sandbox, then the Edit button doesn’t appear in the Structure work area.

3. If prompted, select a context layer to determine the scope of users that your changes affect.

After you complete your changes, you can preview and test the changes, and then publish the sandbox to make your changes available to users.

Creating Groups and Page Entries

To create a group or a page entry:

1. On the Navigation Configuration page, click Create, and select Create Group or Create Page Entry.
2. Enter a name for the group or page entry.
3. Search and select an icon for the group or page entry.
4. If you’re creating a page entry, select the group in which you want to place the new page entry.
5. Select Yes, No, or EL Expression in the Show on Navigator field:
   - Yes: The group or page entry appears on the Navigator. It can also appear on the springboard, depending on what you select in step 6.
   - No: The group or page entry doesn’t appear on the Navigator and springboard.
   - EL expression: The evaluation of the EL expression decides whether the group or page entry will appear on the Navigator and springboard.
6. If you have selected EL Expression for the Show on Navigator field, click the Edit icon next to the Show on Navigator drop-down list, and enter a value or expression. Don’t include spaces or double quotes in the EL expression.
7. For a page entry, if you have set the Show on Navigator field to Yes or EL Expression, then set the Show on Springboard field. The value of this field is evaluated to determine if the page entry will actually appear on the springboard.
   - Yes: The page entry appears on the springboard.
   - No: The page entry doesn’t appear on the springboard.
   - EL expression: The evaluation of the EL expression decides whether the page entry will appear on the springboard.
8. If you’re creating a group, then click Save and Close. If you’re creating a page entry, then perform steps 8 to 10 instead of this step.
9. Select any of the following link types for the page entry:
   - An application page.
   - A dynamic URL of an external page (outside your application) where the host, port, or context root might change.
   - A Static URL of an external page (outside your application) where the host, port, or context root doesn’t change.
10. Based on the link type, specify the required details to configure the link.
11. Click Save and Close.
12. Verify that the new group or page entry that you created appears on the Navigator menu in the sandbox, and then publish the sandbox to make your changes available to users.
Configuring Links for Page Entries: Procedure

While creating a page entry or editing a page entry that was already created for the Navigator and springboard, you can determine what the page entry links to.

Use the Create Page Entry or Edit Page Entry page to link a page entry to any of the following link types:

- Your application page.
- A dynamic URL of an external page (outside your application) where the host, port, or context root might change.
  
  You can determine the host and port details, which a dynamic URL starts with, from a lookup based on the application name.
- A static URL of an external page (outside your application) where the host, port, or context root doesn’t change. Static URLs don’t require lookups.
- A secure token URL of a partner application, to which secure tokens are added. Tokens contain identity and security information about users, and can be used to authenticate them without additional authentication requirements, for example, user name and password.

Linking to Application Pages

To link a page entry to one of your application pages:

1. Select the Application Page link type.
2. Enter the focus view ID of the target page.
3. Select the name of the web application.

You had entered this application name while creating this third party application using the Setup and Maintenance work area.

4. For secure access to the target application page from the page entry, provide the secured resource name and the name of the policy store's application stripe. An example of a secured resource name is oracle.apps.view.pageDefs.CaseList_Form_Attach_UIShellPagePageDef. When a user clicks the link, the application checks the secured resource and the Lightweight Directory Access Protocol (LDAP) policy store. Then, the application determines whether the user has the privilege to view the page.

You can get the application stripe from the jps.policystore.applicationid parameter in the application's weblogic-application.xml file. Examples of application stripes are crm, fscm, and hcm.

5. If the page takes parameters, then you can enter a semicolon-delimited string of name=value pairs (for example, org=m1;context=s1) in the Page Parameters List field.

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. An application page may display or act differently based on the parameters that are passed in. For example, if you’re opening a page from one group on the springboard or Navigator, the parameter might be set to status=Open. Whereas, if you’re opening the page from another group, the parameter might be set to status=Closed.

Tip: If there is another page entry that links to the same application page, then you can enter the same details for all fields, except parameters. On the Navigation Configuration page, open the existing page to view the details, such as Focus View ID, Web Application, and Secured Resource Name. Then, fill in the fields for the new page entry on the Create Page Entry or Edit Page Entry page.
Linking to Dynamic URLs

You can link a page entry to an external website or application that has a frequently changing host, port, or context root. Instead of updating the link to each application, you can update the details of the web application in the topology registration. This change affects all page entries that contain dynamic links pointing to that web application.

For example, suppose you want to link to a test version of an application. So, you use the dynamic URL link type. When you move the application from test to production environment, just change the host and port details of the web application in the topology registration. This change affects all page entries that contain dynamic links pointing to the web application.

As a prerequisite, use the Register Enterprise Applications task to register the web application in the topology.

To link a page entry to a dynamic URL:

1. Select the Dynamic URL link type.
2. Specify the name of the web application and the destination for web application.

For example, suppose you want to link to a complete URL: http://example:9011/myApp/faces/Page1.

Then:

- The name of the web application added to topology is: myApp (the value that would eventually appear in the web application list). The protocol host, port, and context root values of the URL are: http://example:9011/myApp.
- The destination for the web application is: /faces/Page1.

After linking a page entry to a dynamic URL, when you click the page entry, the target page opens in a new browser window or tab.

Linking to Static URLs

You can link a page entry to an external website or application that has a constant host, port, or context root.

To link a page entry to a static URL:

1. Select the Static URL link type.
2. Enter the URL destination. The URL must start with http:// or https://.

For example, you can use a static URL to link to http://www.oracle.com.

Linking to Static URLs with Secure Destinations

To link a page entry to a secure token URL of a partner application (that is, outside your application):

1. Select the Static URL link type.
2. Select Secure Destination.
3. Select the name of the web application.
4. Enter the destination for the web application. An HTTPS protocol is required to access the application.
5. Enter the name of the secure token. All secure tokens have a predefined lifetime, and they expire after that duration. So, users must refresh the page to regenerate the tokens.

The application validates the secure token and uses it to authenticate web services within the end user context. Using this mode of modified access, a partner can directly perform an action or display information to the specific user without any additional authentication.
Managing Groups and Page Entries for the Navigator and Springboard: Procedure

Use the Navigation Configuration page to edit and reorder the existing groups and page entries.

Prerequisites

Following are the prerequisites:

1. From the Navigator menu, select **Configuration > Structure**.
2. Activate a sandbox. If you're not in an active sandbox, click **Edit** in the Structure work area. You're prompted to activate a sandbox.
   - **Tip**: If you're already in an active sandbox, then the **Edit** button doesn't appear in the Structure work area.
3. If prompted, select a context layer to determine the scope of users that your changes affect.

After you complete your changes, you can preview and test the changes, and then publish the sandbox to make your changes available to users.

Editing Groups and Page Entries

Follow these steps:

1. On the Navigation Configuration page, click the name link for the group or page entry.
   - **Tip**: You can use the search panel on the Navigation Configuration page to find the group or page entry you want to edit.

   If you get a **webApp value not defined** error message on clicking a group or page entry, verify whether the application is in the topology tables. For more information, refer to the Oracle Fusion Applications Administrator's Guide.

2. On the Edit Page Entry page or the Edit Group page, make the required changes.
3. Click **Save and Close**.

You can make the following changes to a group or page entry:

- Rename a group or page entry.

  - **Note**: If a group or page entry was created using a different tool, then you can't change its name using the Navigation Configuration page.

- Change the icon for a group or page entry.
  - If a page entry was created using a different tool, then you can't change its icon using the Navigation Configuration page.
• For a page entry, change the group in which the page entry is placed.
• Change the **Show on Navigator** property for the group or page entry.
• Change the **Show on Springboard** property for the page entry.
• For an administrator-defined page entry, change the settings for link configuration.
• Delete page entries that were created using the Navigation Configuration page.
• For a predefined page entry, use **Create Duplicate** on the Edit Page Entry page to create a duplicate page entry. You can then edit the duplicate page entry as you want, for example, you can place the duplicate page entry in a different group or at the top level.
• For groups with associated quick actions, use the **Quick Actions** tab to create more quick actions or make changes to the existing ones, such as rename, show or hide, and reorder them.

**Editing Page Entries with Tabs**

Some pages (for example, Security Console) have tabs. Each tab is a task flow. To edit tabs, click the **Tabs** tab on the Edit Page Entry page. You can make the following changes:

• Click the tab name to rename it.
• Click the tab icon to search and select another icon for the tab.
• Click the **Visible** field for the tab, and change the option to show or hide the tab.
• Use the **Move Up** and **Move Down** icons to adjust the relative position of the tabs within the page.

**Editing Page Entries with Panel Tabs**

Some pages have panel tabs. To edit panel tabs, click the **Panel Tabs** tab on the Edit Page Entry page. You can make the following changes:

• Click the panel tab name to rename it.
• Click the panel tab icon to search and select another icon for the panel tab.
• Click the **Visible** field for the panel tab, and change the option to show or hide the tab.
• Use the **Move Up** and **Move Down** icons to adjust the relative position of the panel tabs within the page.

**Reordering Groups and Page Entries**

Use the **Move Up** and **Move Down** icons on the Navigation Configuration page to reorder groups and page entries. For page entries, you can use the **Move To** icon to move page entries to different groups or to the top level.

After completing your changes, verify and test the changes in the sandbox, and then publish the sandbox to make your changes available to users.

**Related Topics**

• Quick Actions: Explained
• Configuring Quick Actions: Procedure
FAQs for Navigator and Springboard Configuration

Why can't I edit the Structure page entry or the Tools group?
While configuring the Navigator and springboard, you can't:

• Override the Navigator setting for the Structure page entry and Tools group. The default setting is Yes, so the Structure page entry and Tools group always appear on the Navigator and springboard.

• Move the Structure page entry to a different group or to the top level. The default group is Configuration, so the Structure page entry always appear in the Configuration group.

Why are some springboard icons and Navigator menu items not displayed on the springboard or Navigator?
A springboard icon or a Navigator menu item may be hidden due to any of the following reasons:

• The administrator hasn’t enabled the offering associated with the group or page entry.

• You may not have security privileges to access or view the group or page entry. If the group or page entry isn’t available on both, the Navigator menu and springboard, then that means you don’t have the appropriate privileges.

• The administrator has hidden the group or page entry from the springboard or the Navigator menu using the Structure page. To review the visibility settings, from the Navigator menu, select Configuration > Structure.

Related Topics

• Enabling Offerings: Explained

• Configuring Offerings: Procedure

• Why are springboard icons appearing at the top level instead of appearing within a folder?

Configuring Home Page Navigation: Procedure

Use the Home Configuration page to configure the icons for infolet pages or other configurable pages in the page control on the home page.

Prerequisites
Following are the prerequisites:

1. From the Navigator menu, select Configuration > Structure.
2. Click the Home Configuration tab.
3. Activate a sandbox. If you’re not in an active sandbox, click Edit in the Structure work area. You’re prompted to activate a sandbox.

Tip: If you’re already in an active sandbox, then the Edit button doesn’t appear in the Structure work area.
If prompted, select a context layer to determine the scope of users that your changes affect. After you complete your changes, you can preview and test the changes, and then publish the sandbox to make your changes available to users.

Defining Settings

You can rename icons for infolet pages and other configurable pages in the page control, change their visibility settings, and reorder them. On the Home Configuration page, you can:

- Click the infolet name or any other configurable page name to rename it.
- Click the **Visible** field for an infolet or any other configurable page to change its visibility setting. You can show or hide the icon for these pages in the page control on the home page. You can select one of the following options:
  - **Yes**: The icon appears in the page control.
  - **No**: The icon doesn’t appear in the page control.
  - **EL expression**: The evaluation of the EL expression decides whether the icon appears in the page control.

- Click the **Default View** field for an available configurable page to specify whether the page should be set as the default home view. You can select one of the following options:
  - **Yes**: The page is set as the default home view.
  - **No**: The page isn’t set as the default home view.
  - **EL expression**: The evaluation of the EL expression decides whether the page is set as the default home view.

  **Note**: Only specific configurable pages, such as Quick Actions, are available for you to set as the default home view. When you click the **Default View** field for such pages, you get the options to select **Yes**, **No**, or **EL Expression**. These options aren’t available for other pages that you can’t set as the default home view.

- Use the **Move Up** and **Move Down** icons to adjust the relative positions of the icons for the infolet pages or other configurable pages in the page control on the home page.

You can use profile options to define settings for the filmstrip, which you can find above all simplified pages:

- To enable users to use the filmstrip, set the **Springboard Strip Enabled** profile option (**FND_USE_FILMSTRIP**) to **Yes**.
- If the **FND_USE_FILMSTRIP** profile option is set to **Yes**, then you can display the filmstrip as expanded by default. To do so, set the **Springboard Strip Expanded** profile option (**FND_EXPAND_FILMSTRIP**) to **Yes**. A user can still collapse or expand the strip on any page, and when done, this profile option is set by default for subsequent sessions of that user.

**Related Topics**

- Setting Profile Option Values: Procedure
- Defining Home Page Display Settings: Procedure
- Setting Up Sandboxes: Procedure
Using EL Expressions for Configuring Navigation: Examples

You can use EL expressions to configure navigation in the application such as, to show or hide the navigator menu items, and the icons for infolet pages in the page control on the home page. The following scenario shows how you might use EL expressions.

Using EL Expressions for Configuring the Navigator Menu Items

Use EL expressions to specify whether a group or page entry should appear on the Navigator and springboard. The evaluation of the EL expression decides whether the menu items are displayed for a user.

The following table provides examples of how you can use EL Expressions to show or hide groups and page entries from the Navigator menu or springboard for specific users:

<table>
<thead>
<tr>
<th>Who can see the group or page entry</th>
<th>EL Expression and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only users having any of the specific roles</td>
<td>#{securityContext. userInRole['&lt;RoleName&gt;']}</td>
</tr>
<tr>
<td></td>
<td>#{securityContext. userInRole['ORAFND APPLICATION_ADMINISTRATOR_ JOBORA_ PER_ EMPLOYEE_ABSTRACT']}</td>
</tr>
<tr>
<td>Only users not having any of the specific roles</td>
<td>#{!(securityContext. userInRole['&lt;RoleName&gt;'])}</td>
</tr>
<tr>
<td></td>
<td>#{!(securityContext. userInRole['ORAFND APPLICATION_ADMINISTRATOR_ JOBORA_ PER_ EMPLOYEE_ABSTRACT'])}</td>
</tr>
<tr>
<td>Only users having all of the specific roles</td>
<td>#{securityContext. userInAllRoles['&lt;RoleName&gt;']}</td>
</tr>
<tr>
<td></td>
<td>#{securityContext. userInAllRoles['ORAFND APPLICATION_ADMINISTRATOR_ JOBORA_ PER_ EMPLOYEE_ABSTRACT']}</td>
</tr>
<tr>
<td>Only users not having all of the specific roles</td>
<td>#{!(securityContext. userInAllRoles['&lt;RoleName&gt;'])}</td>
</tr>
<tr>
<td></td>
<td>#{!(securityContext. userInAllRoles['ORAFND APPLICATION_ADMINISTRATOR_ JOBORA_ PER_ EMPLOYEE_ABSTRACT'])}</td>
</tr>
<tr>
<td>Only users having access to specific resources, for example, the Social work area</td>
<td>#{securityContext. userGrantedResource['resourceTypeFNDResourceType;resourceName FNDLaunch Social_Menu;action= launch']}</td>
</tr>
</tbody>
</table>

Don't include spaces or double quotes in EL expressions.

**Note:** Use an EL expression that can be evaluated from any page because the Navigator menu is used on all pages. So, don’t use product-specific EL expressions.
Help Content Management

Managing Help Content: Overview

You can add and edit help files in Applications Help and help windows, as well as help text that appears on UI elements. You can also tailor the pages in the Getting Started work area.

**Note:** To allow help file and Getting Started changes, you or your implementor must enable the Help Content Management feature in your offering.

Help Files

A help file, such as an FAQ or example, provides information about a specific topic. You can find help files in:

- **Help windows:** Which can contain informational text as well as links to help files. Click a help icon next to, for example, a page or section title to open the help window and find information about that page or section.
- **Applications Help:** The help site which contains all help files, including help files found in help windows. To open Applications Help at any time, click your user image or name in the global header and select **Applications Help**.

You have many options to manage help files, including:

- Creating help files with your own content and adding them to help windows
- Editing the content in predefined help files

Getting Started Pages

The Getting Started work area contains a set of pages that provides information for new users. To open this work area, select **Getting Started** from the Navigator. You can modify predefined pages and create pages with your own content.

Embedded Help

Help text might appear when you hover over or click certain UI elements on the page. For example, hint text can appear when you put your cursor in a specific field. The help text you see on the page is called embedded help. To manage embedded help, you can, for example:

- Edit the help text that appears for a specific check box.
- Add help text that appears when users hover over a specific tab.

Related Topics

- Opting in to Features: Procedure
Who can create, edit, and manage help?

Users with the Manage Help Content (ATK_CUSTOMIZE_HELP_TOPICS_PRIV) privilege can create and edit:

- Help in Applications Help and help windows
- Pages in the Getting Started work area

This privilege is assigned by default to the administrators for product families. Your security administrator can define which users have job roles with this privilege.

Help Files Management

Overview

If you have the appropriate roles, then you can add and edit help files in Applications Help and help windows (which users open using help icons on the pages they work in). You can also determine which help files appear in which help windows, and which product family page a file belongs to in Applications Help.

What you can do to a help file depends on whether it’s added or predefined.

- **Added:**
  - Create, duplicate, edit, and delete
  - Set status (Active or Inactive)

- **Predefined:**
  - Duplicate
  - Edit (which is really creating a copy of the predefined file that you can modify)
  - Set status

Navigation

This table describes where to go to manage help.

<table>
<thead>
<tr>
<th>Help Management Task</th>
<th>Navigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create or edit help for a specific help window.</td>
<td>Click the <strong>Manage Help Content</strong> link in the help window.</td>
</tr>
<tr>
<td>Edit any help file, including glossary terms.</td>
<td>Open the file in Applications Help and click the <strong>Edit</strong> link.</td>
</tr>
<tr>
<td>Perform any help file management task, including managing a set of help, such as all help for a product.</td>
<td>Go to Applications Help and click your user image or name in the global header to select Manage Help Content.</td>
</tr>
</tbody>
</table>
### Help Management Task

Make a copy of all added help for testing, migration, or other purposes.

### Navigation

Create a configuration package then use the export and import in the Setup and Maintenance work area.

- The configuration package must use a source implementation project that contains the Define Help Configuration task list.
- Select the following objects to export:
  - Help Configuration
  - Help Topic

---

**Related Topics**

- Implementation Project Based Export and Import: Explained

---

## Source Types for Help Files: Explained

You have many options to determine the content of a help file, for example by uploading a local file, specifying a URL, or typing in the text. When you create help files, you first select a value for the **Source** list, to indicate how to provide your content.

### Desktop File

Upload a file of any type from your computer.

### Oracle User Productivity Kit

Identify the Oracle User Productivity topic to use as help.

### Text

Use a rich text editor to enter the content of the help file.

### URL

Enter the full URL to a Web site or a file of any type.

### Video URL

For a video to play in the help windows assigned to the help file, enter the full **URL** to the video on YouTube or elsewhere.

---

## Help Types: Explained

Applications Help has many types of help content: examples, FAQs, glossary terms, help topics, PDF guides, and videos.

### Example

Examples can provide:

- Real use cases to illustrate how and when to do something
• Scenarios to explain abstract concepts

Worked examples show exactly what you do to achieve a specific result. They emphasize decisions that you make and values that you enter.

FAQ
FAQs, or frequently asked questions, provide brief answers to questions that you might have about a task or page. For example, they can explain:

• What a term means
• Why something happened
• How you can perform an action
• What happens if you perform the action

Glossary Term
Glossary terms provide definitions for words or phrases used in help content. In a help file, terms that have glossary definitions are presented as links. When you see terms underlined with dots, you can hover over the term to see its definition. To see the whole glossary, select Glossary from the Navigator menu in Applications Help.

Help Topic
Help topics can:

• Explain key concepts
• Tell you the steps to follow to perform tasks
• Help you make decisions by explaining points to consider or describing the options you have
• Show you how application components work together
• Provide reference, overview, and other information

PDF Guide
PDF guides provide information in a book format. The guides in Applications Help contain content that you usually can’t find in other help types.

✏ Note: Most of the examples, FAQs, and help topics in Applications Help are also in guides. To see these guides, select Documentation Library from the Navigator menu in Applications Help.

Video
Videos, or tutorials, show you how to complete a short task or part of a task. Videos can also give you an overview of complex dashboards and work areas.

Page or Section Values: Explained
The Page or Section value represents where users can click a help icon to open a help window that can contain links to help files. Use this value on the Manage Help Content page in Applications Help to:

• Search for help that appears in a specific help window.
• Add help files to the help window.
Where Help Windows Are Found

In most cases, the page or section value corresponds to a page or section header that has a help window. Help windows are also available:

- On specific tabs, dialog boxes (windows), panel tabs, or dashboard regions
- In the Setup and Maintenance work area, for example within the table of tasks on the Setup page
- In the Offerings work area, for example within the table on the Edit Features page
- In the Learn More column in the New Features work area

How to Read the Values

The page or section value reflects the logical navigation to the help window. For example, Process Details window, Output tab doesn’t mean that the help window is in two different places. It’s in the Output tab within the Process Details dialog box.

For objects in the Setup and Maintenance, Offerings, and New Features work areas, the page or section value reflects the object that the help window is for. These values end with offering, functional area, feature, task list, or task, for example, Set Help Options task.

When the Same Help Window Is On Multiple Pages

In some cases, a page or section value does represent a single help window that appears on multiple pages. If the value is:

- For example Create and Edit Message pages, then the same help window is on both the Create Message and Edit Message pages.
- Just a region name without a page or window name, then the same help window is in a section or dashboard region that appears on multiple pages.

Managing Help in Help Windows

Video

Watch: This video tutorial shows you how to add and edit help in help windows. The content of this video is also covered in text topics.

Procedure

When users click help icons on a page, they open help windows that provide information on how to use the page or part of the page. Help windows have informational text, links to help files, or both. Use the Manage Help Content dialog box to edit those help files, create new files to appear in the help window, or change the order of links in the window. To restrict access to any of those help files, use the Manage Help Content page in Applications Help to assign a help security group to the help file.

Tip: You can also use:
- The User Interface Text tool to edit the informational text that appears in a help window
- Page Composer to hide the More Help link at the bottom of help windows
Opening the Manage Help Content Dialog Box

Use the Manage Help Content dialog box to create and edit help files, or to reorder links, for a help window:

1. Click the help icon to open the help window. If you don’t see any help icons, click your user image or name in the global header and select Show Help Icons. To display help icons by default for all users when they sign in, use the Set Help Options task in the Setup and Maintenance work area.

2. Click Manage Help Content.

Creating Help Files

To create a help file that appears in the help window (and also Applications Help):

1. Open the Manage Help Content dialog box for the help window.
2. Click Create.
3. Select a source and, for any source other than Text, enter information that points to the content of the help file.
4. Select the help type, which corresponds to the Help Type filter in Applications Help.
5. Enter the title, which is the text of the link that appears in the help window.
6. Enter keywords to help users find your file in Applications Help.
7. Enter a description for your file, which users can see in the help window or in Applications Help search results before they open the entire file.
8. If the source for your file is Text, enter the help content in the rich text editor.
9. Optionally select a language, country, or role, which correspond to filters in Applications Help.
10. Save your work.
11. To verify your work, click the title of your new help file in the help window.
12. Click the More link at the end of the description to see the entire help content. If your new file is an FAQ, you can then click the Related FAQs link to see the FAQ in Applications Help along with other, related FAQs.

Editing Help Files

To edit help files in the help window:

1. Open the Manage Help Content dialog box for the help window.
2. Click the title of the help file.

⚠️ Note: If you’re updating a predefined file, then you see the Create Help dialog box because you’re really creating a copy of the predefined file that you can then edit.

3. Change the title or content of the help file as needed, as well as the status.

   Setting the status to Inactive hides the file in the help window and in Applications Help.

4. Open the More Details section if you need to update more aspects of the help file, including fields that correspond to the following filters in Applications Help:
   - Help Type
   - Role
   - Language
   - Country

5. Save your work.
6. To verify your work, reopen the help window.
7. Click the title of the updated help file.
Click the More link at the end of the description to see your changes in the entire help file.

Reverting to Predefined Help Files
When you edit a predefined help file, you're actually creating a new file and inactivating the predefined file. To discard your changes and revert back to the predefined file:

1. Open the Manage Help Content dialog box for the help window.
2. Click the title of new help file that was created when you edited a predefined file.
3. In the Edit Help dialog box, click Delete.
4. Reopen the Manage Help Content dialog box for the help window.
5. Click the title of the predefined help file, which is appended with (Inactive).
6. Select Active as the status.
7. Save your work.

Adding Links to Help Windows
To add links in the help window to any help file that was already created:

1. In Applications Help, click your user name in the global header and select Manage Help Content.
2. In the search, select the page or section that corresponds to the help window.
3. Click Search.
4. In the search results toolbar, click Select and Add to Help Location.
5. Search for the help file that was already created, select it, and click Apply.
6. Repeat the previous step for all the help files you want to add.
7. Click Done.

Related Topics
• How can the icons for help windows be shown by default?

Assigning Help Locations: Procedure
To determine which help windows your help file appears in, assign the appropriate help locations to the file. You also use help locations to define where help files appear in the Task and Product filters in Applications Help, as well as which product family tabs the files belong to. Help locations include:

• Task hierarchy for the Task filter and product family tabs
• Product hierarchy for the Product filter and product family tabs
• Page or section values for help windows

Help locations are available only on the Manage Help Content page in Applications Help, not the Manage Help Content dialog box from help windows. When you create help from the help window, the help files are automatically assigned to the help window and task hierarchies associated with the window.

Determining Where Help Appears in the Task Filter, Product Filter, and Product Family Tabs
Enter at least one hierarchy, as many as you need:

1. In Applications Help, click your user name in the global header and select Manage Help Content.
2. Find the help file to edit or duplicate, or click Create.
3. Enter or update the general information for the help file.
4. In the Help Location section, add a row if there isn’t already a blank one, or edit an existing row.
5. Select Task or Product for the hierarchy type.
6. Select nodes for as many levels of the hierarchy as you need, starting with level 1. If the Task hierarchy nodes you’re assigning are at level 4 or lower, then click the Details icon to assign nodes.
7. Add more rows as needed.
8. Save your work.

Selecting Help Windows
Every page or section value is associated with a specific node in the Task hierarchy. When users click More Help from a help window, they get all the help files that are assigned to the same Task node as the page or section value.

To determine the help windows that a help file appears in:
1. Enter the Task hierarchy that’s associated with the help window, as described in the previous procedure, to narrow down the available page or section values.

   Note: For any help window for specific objects in the Setup and Maintenance, Offerings, and New Features work areas, select this hierarchy:
   - Hierarchy: Task
   - Level 1: Functional Setup
   Such objects include offerings, functional areas, features, task lists, and tasks.
2. Select the page or section in the same row.

If you know the exact page or section you want, then you can select the value without entering a Task hierarchy. The associated hierarchy automatically fills in the rest of the row.

Adding Help to the Getting Started Panel in Applications Help:

Procedure
In Applications Help, users can use the springboard or filmstrip to open landing pages for specified product families. Every product family page has a Getting Started panel, which contains guides and videos to help new users. Users can also find help topics for new users in specific roles. You can add your own help to the Getting Started panel for any product family.

Adding Help to the Panel for a Product Family
Follow these steps:

1. Click your user name in the global header in Applications Help, and select Manage Help Content.
2. Create or edit the help file you want to add to the Getting Started panel.
3. In the General Information section of the Create Help or Edit Help page, click the Getting started check box.
4. Make sure that the help file has either or both of the following:
   - Help type of PDF guide or Video
   - One or more roles assigned
5. In the Help Location section, enter at least one row with Product in the Hierarchy column.
6. In the Level 1 column, select the product family for which you want to add your help file to the Getting Started panel. Optionally select a product in the Level 2 column.
7. Save your work.
Links in Help Files: Points to Consider

When you create or edit help, follow best practices when you include links to help files or other content. If you’re duplicating a predefined help file, then you may see existing links. The types of links that you can work with include:

- Related help links
- Standard hypertext links
- Links to documentation library content
- Glossary term links

For all link types, except the standard hypertext links, you must create or edit help with a Text or Desktop File source type. For standard hypertext links, the source type can also be URL.

Related Help Links

Related help is the section at the end of help files that contains links to other help files. The syntax for related help contains a comma-separated list of title IDs that represent help files, for example:

OfaRelatedTopics(CREATE_AUTOMATIC_CRITERIA_S_0000, JOURNAL_ENTRIES_HOW_THEY_RE_RECORDERE_0000)

In this example, the help file has two links to related help.

- To remove all related help, delete this code.
- To remove individual links, delete only title IDs (for example, CREATE_AUTOMATIC_POSTING_CRITERIA_S_0000).
- To replace existing links or add new links, retain the code syntax and enter the appropriate title IDs. To find title IDs, search for the help files on the Manage Help Content page. Show the Title ID column in the search results if the column is hidden.

Standard Hypertext Links

You can create standard hypertext links to any file or Web site as long as you make sure that the links are valid and stable. These links can appear anywhere in the body of your help file as long as they come before any related help links.

If you’re working on a help file with the Text source type:

1. In the Help Content section of the Create or Edit Help page, highlight what you want to use as link text.
2. Click the Add Link icon.
3. Enter the full URL, for example http://www.oracle.com.

***Tip:** To find the URL for a help file that you want to link to, open that help file in Applications Help, and click the Bookmark link.

Links to Documentation Library Content

The syntax for links to HTML files in documentation libraries is:

```html
```

In this example, WCSUG4636 is the anchor ID and Understanding Tags is the link text. You can:

- Change the link by replacing the existing anchor ID, editing the link text, or both.
- Remove the link by deleting all the code for it.
• Create links to documentation library content by following the same syntax. These links can appear anywhere in the body of your help file as long as they come before any related help links.

**Note:** To ensure that you’re linking to a supported documentation library, enter anchor IDs only from documentation libraries that are linked from predefined help.

**Glossary Term Links**

Glossary term links provide definitions in a note box when users hover over the term in help files. The syntax for glossary term links is:

```html
OgaGlossaryTerm("accounting period", ACCOUNTING_PERIOD_0001)
```

In this example, `accounting period` is the link text, or glossary term, and `ACCOUNTING_PERIOD_001` is the identifier, or title ID.

• To remove the link but retain the text, delete all the code except the term itself.

• To add glossary term links, you must follow the link syntax and use the correct title ID for the glossary term. To find title IDs, search for the glossary terms on the Manage Help Content page. Show the **Title ID** column in the search results if the column is hidden.

If your help file has the Desktop File source type, then make sure before uploading that the quotes around glossary terms are actual quotation marks in raw HTML, not `&QUOT;`. Otherwise, quotation marks will appear when users view the help file.

**Modifying PDF Guides in Help: Worked Example**

This example demonstrates how to modify a PDF guide that came with Applications Help. This guide is currently not available from any help window.

The following table summarizes key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What changes do you need to make to the guide?</td>
<td>Change the title of the guide and hide all the content that’s not about a particular subject</td>
</tr>
<tr>
<td>Should the modified guide appear in any help window?</td>
<td>Yes, the help window for the Manage Administrator Profile Values page</td>
</tr>
<tr>
<td>Which products and tasks should the modified guide be assigned to?</td>
<td>Same as the original guide, plus the task associated with the help window</td>
</tr>
<tr>
<td>Do you want to limit access to the modified guide?</td>
<td>No, same as the original guide</td>
</tr>
<tr>
<td>Do you want to tag the modified guide with a role for the Role filter?</td>
<td>Yes, the Application Administrator or Implementor role</td>
</tr>
</tbody>
</table>

For this scenario:

1. Edit a copy of the original PDF guide.
2. Edit the help file to replace the original PDF guide with your modified guide.
Copying and Editing the PDF Guide

1. Open the original PDF guide in Applications Help and save a copy to your desktop. Leave the help file for the guide open.
2. Using a PDF editor application, change the title of the guide wherever it appears. Delete the content you want to hide from users.

Replacing the Original PDF Guide

1. In the help file that you still have open for the original PDF guide, click the Edit link. You automatically create a copy of the predefined help file that you can edit.
2. On the Create Help page, use the default values except where indicated.
3. Update the title to the name that you want to display to users.
4. In the File Name field, browse for and select your modified guide.
5. Delete any keywords or parts of the description relevant to the content you removed from the PDF guide.
6. From the Roles list, select Application Administrator or Implementor.
7. Add a row in the Help Location table.
8. Click the icon in the Details column for the new row, and complete the fields, as shown in this table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Task</td>
</tr>
<tr>
<td>Level 1</td>
<td>Functional Setup</td>
</tr>
<tr>
<td>Level 2</td>
<td>Perform Functional Setup</td>
</tr>
<tr>
<td>Level 3</td>
<td>Set Application Options</td>
</tr>
<tr>
<td>Level 4</td>
<td>Define Profiles</td>
</tr>
</tbody>
</table>

9. Click OK.
10. Select Manage Profile Option Values page in the Page or Section column.
11. Click Save and Close. The help file for the original PDF guide is automatically set to inactive.

Adding User Productivity Kit Content to Help: Worked Example

This example demonstrates how to add an Oracle User Productivity Kit topic as a video help file in Applications Help.

Note: Your topic must be made with User Productivity Kit 3.6.1 or later to be added as help.

The following table summarizes key decisions for this scenario.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>What User Productivity Kit content do you want to add to help?</td>
<td>One topic from a module with five topics</td>
</tr>
<tr>
<td>Should the topic appear in any help window?</td>
<td>Yes, the help window for the Overview page in the Scheduled Processes work area</td>
</tr>
</tbody>
</table>
Decisions to Consider | In This Example
--- | ---

For this scenario:

1. Generate a report of User Productivity Kit document IDs, to identify the User Productivity Kit topic when you create your help file.
2. Publish the module as a player package.
3. Create a help file for the User Productivity Kit topic.

Generating a User Productivity Kit Document ID Report

1. In the User Productivity Kit Developer, select **Details View**.
2. Right-click any column header, for example Name, and select **Column Chooser**.
3. In the Column Chooser dialog box, move the Document ID column header after the Name column.
4. Close the Column Chooser dialog box.
5. From the File menu, select to print, and save the output as a Microsoft Excel file to your desktop.

Publishing the Module as a Player Package

1. From the Developer, make sure that the topic that you want to add as a help file has the See It play mode. The topic can also have other modes, but only the See It mode is included in your help file.
2. Publish the module, specifying any location for the output and selecting to publish the selection only.
3. In the Formats section of the Publish Content window, select the **Player** check box in the **Deployment** check box group.
4. In the Player section, select the **Include HTML Web Site** check box. Your help file includes a text-only version of the User Productivity Kit topic.
5. Finish the publishing process, after optionally setting other options.
6. Go to where you specified the output to be generated.
7. In the Publishing Content folder, copy the PlayerPackage folder and add it to the Web server where you store User Productivity Kit content.

Creating a Help File for the Topic

1. Open the Scheduled Processes work area.
2. Open the help window for the Overview page, and click **Manage Help Content**.
3. Click **Create**.
4. In the Create Help dialog box, complete the fields as shown in this table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>Oracle User Productivity Kit</td>
</tr>
<tr>
<td><strong>File Location</strong></td>
<td>The full URL of the player package folder on the Web server, for example, http://&lt;your domain&gt;.com/MyContent/PlayerPackage</td>
</tr>
<tr>
<td><strong>Document ID</strong></td>
<td>The document ID of the User Productivity Kit topic to add to the help window on the Scheduled Processes Overview page. Copy and paste this ID from the Microsoft Excel file that you generated earlier.</td>
</tr>
<tr>
<td><strong>Help Type</strong></td>
<td>Video</td>
</tr>
<tr>
<td><strong>Topic Title</strong></td>
<td>The name of the User Productivity Kit topic.</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Keywords</td>
<td>Terms relevant to the topic.</td>
</tr>
<tr>
<td>Description</td>
<td>Summary of the topic.</td>
</tr>
</tbody>
</table>

5. Click **Save and Close**.

### Editing Glossary Terms: Procedure

You can edit the glossary terms in the Applications Help glossary. These are the same terms that users might find as links in help topics. The links provide definitions when users hover over the terms.

**Procedure**

Follow these steps:

1. In the global header of Applications Help, select **Glossary** from the Navigator.
2. Find the glossary term.
3. Click **Edit**.
4. Update the topic title (the term) or definition as needed.
5. Optionally change the assigned hierarchies.
   - The last node of each hierarchy appears after the term’s definition in the glossary.
   - Glossary terms don’t appear in the Task or Product filter.
6. Save your work.

### Predefined Glossary Terms

When you edit a predefined glossary term:

- You’re actually creating a copy, and the original term becomes inactive.
- The glossary in Applications Help shows your copied version.
- In help files, existing links to the glossary term automatically point to your version.

> **Note:** If you later inactivate your glossary term, make sure to activate the original term so that the links still work.

### FAQs for Help Files Management

**What happens when I edit predefined help?**

You’re actually creating a new help file based on the predefined file. The new version replaces the original, which becomes inactive and hidden from users. You can display both versions by reactivating the original file.

**How can I add a YouTube video to a help file?**

Create a help file using the URL to the YouTube video.

1. Find the video in YouTube.
2. Click the **Share** button.
3. Click the **Embed** button.
4. Copy the URL within the embed code, for example, `http://www.youtube.com/embed/<unique ID>`.
5. Open the Create Help page from a help window or from the help site.
6. Select **URL** as the source, or **Video URL** if you want the video to play within help windows.
7. Paste your copied URL in the **URL** field.

**Tip:** If you selected **Video URL**, change your pasted URL to start with `https` so that users don’t need to adjust browser settings to play the video in the help window.

8. Enter other information, and click **Save and Close**.

**How can I restrict access to specific help files?**

You must create or edit help from the Manage Help Content page.

1. In Applications Help, click your user name in the global header and select **Manage Help Content**.
2. As you create or edit a help file, select a help security group, which represents a set of roles that have access to the help.

   The predefined groups are:
   - **Unsecured**: Anyone can view the help.
   - **Secured**: All internal employees and contingent workers have access (unless this group was edited).

   If you don’t see the **Security Group** field, then you or an administrator can enable this feature in the Offerings work area.
3. Save your work.

**Related Topics**

- Creating Help Security Groups: Worked Example
- Setting Up for Creating and Editing Help: Procedure

**What’s the difference between assigning a role and a security group to a help file?**

When users filter or browse for help files by role in Applications Help, they get the help files tagged with the role. The help security group hides the help file completely from users who don’t have the roles defined in the group.

**Why can’t users find my help files in their search results?**

If you created or edited those help files recently, they might not be indexed yet for the search in Applications Help. The indexing process runs on a defined schedule. Users can still find your help files by browsing, for example using the Task or Product filter.

**What happens to my help files after a release update for Applications Help?**

Nothing happens to the help files you create or edit. Upgrades affect only predefined help files, active or inactive.

Take a look at any inactive predefined file that’s updated to see if you want to:

- Activate the updated version.
- Make similar edits to your version of that file, if any.
How can I change the background image on the Applications Help home page?

In the Setup and Maintenance work area, open the Set Help Options task, and upload your own image in the Help Site Appearance section. If you don’t see the Help Site Appearance section, then check with your implementor about enabling the Help Content Management feature.

Getting Started Pages Management

Overview

The Getting Started work area provides pages of information to introduce new users to the application. If you have multiple cloud services, you get a separate set of pages for each service. You can edit the predefined pages or add your own, so that users get content specific to your organization.

Access

Open the Getting Started work area, and, if you have multiple sets of pages, select the set to modify. You then see the Edit Getting Started link if you have the appropriate roles.

Key Tasks

You can:

- Create and edit pages using a rich text or HTML source code editor
- Reorder the pages
- Activate or inactivate any page

Tip: To hide the link to a set of Getting Started pages, inactivate all pages within that set. Users with access to edit Getting Started pages can still see the link, but everyone else can’t. If you inactivate all sets except one, then users land on the first page of the active set when they open the Getting Started work area.

- Delete added pages (not predefined ones)

How can I add a YouTube video to a Getting Started page?

Create or edit a page in the Getting Started work area, and include a piece of code from YouTube.

1. Find the video in YouTube.
2. Click the Share button.
3. Click the Embed button.
4. Copy everything in the text box.
5. Back in the application, open the Getting Started work area.
6. If you have multiple sets of Getting Started pages, select the set you want to add the video to.
7. Click the Edit Getting Started link.
8. Open an existing Getting Started page or create a new one.
9. Click **Source Code Editing Mode** in the toolbar.
10. Paste in the code you copied from YouTube.
11. Click **Save and Close**.

Embedded Help Management

Managing Help That Appears on the Page: Overview

You can create, edit, or delete help that you see on the page, for example hints for buttons or text in help windows. There are different types of such embedded help. Embedded help doesn’t include help that you open using links in help windows, or help that you find in Applications Help.

Creating, Editing, or Deleting Embedded Help

- Use Page Composer to edit, create, or delete hint text that appears when you hover over or click certain UI elements, for example buttons, icons, fields, and check boxes. Open the properties of the UI element to define the help text in the shortDesc field. Not all types of embedded help can be created or modified using Page Composer, for example help windows.
- Use the User Interface Text tool to edit the text for any type of embedded help, including informational text in help windows. You usually use this tool to make bulk changes, for example to change a phrase wherever it appears in any UI label, embedded help, messages, and so on.

Related Topics

- **Modifying Simplified Pages Using Page Composer: Procedure**
- **Page Component Properties: Explained**
- **Bulk Text Modifications: Explained**
14 Extending Activities and Sales Orders

Overview
Learn about the Activity and Sales Order objects, and how to extend those objects using Application Composer.

Extending Simplified Pages for Activities: Explained
Use Application Composer to change the items that appear on the simplified set of pages for activities.
You can make changes to the following pages:

- Activities summary list
- Create Task
- Create Appointment
- Create Call Report
- Edit Task
- Edit Appointment
- Edit Call Report
- My Tasks
- My Appointments
- Activities subtab, available from the details pages of other objects, such as opportunity, account, contact, and so on.

For example, you can:

- Hide or show existing fields.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

⚠️ **Note:** To extend the simplified set of pages for activities, use the Activity object in Application Composer.

Activities Summary List
To extend the Activities summary list:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click **My Activities List View**.
Create Task, Create Appointment, and Create Call Report Pages

To extend the Create Task, Create Appointment, and Create Call Report pages:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout for either the Create Task, Create Appointment, or Create Call Report page to create a new layout to edit. Or, edit another existing layout.

   To enhance run time usability for your end users, optionally group custom fields into field groups.

Edit Task, Edit Appointment, and Edit Call Report Pages

The Edit Task, Edit Appointment, and Edit Call Report pages consist of multiple subtabs that display along the left side of the page. You can modify most of these subtabs. You can also add custom subtabs to display records from child or related objects, for example.

To extend the Edit Task, Edit Appointment, and Edit Call Report pages:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout for either the Edit Task, Edit Appointment, or Edit Call Report page to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Task, Edit Appointment, and Edit Call Report pages, select the subtab you want to change first. The following subtabs are available from the Edit Task, Edit Appointment, and Edit Call Report pages. Use the following subtabs to configure which standard and custom fields display at run time.

   ° Summary
   ° Contacts

   Use the Activity Contact child object to add custom fields as needed, before you can select them for display on the subtab.

   ° Resources

   Use the Activity Assignee child object to add custom fields as needed, before you can select them for display on the subtab.

   ° Notes

   Use the Note object in Application Composer to create custom fields as needed, and to extend this subtab.

   On the Simplified Pages tab for the Note object, edit the default creation page layout or duplicate it to create a new layout.

   The Notes list page is not extensible.

Tip: The changes you make to these subtabs are unique across the Edit Task, Edit Appointment, and Edit Call Report pages. For example, the changes you make to the Contacts subtab on the Edit Task page are not reflected on the Contacts subtab on the Edit Appointment page.
My Tasks
To extend the My Tasks page:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click Edit My Tasks Table.

My Appointments
To extend the My Appointments page:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click Edit My Appointments Table.

Embedded Activities Subtab
To extend the Activities subtab, available from the details pages of other objects:

1. Navigate to the Activity object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click Edit Embedded Summary Table.

Extending Simplified Pages for Sales Orders: Explained
Using Application Composer, you can change many items that appear on the simplified set of pages for sales orders. The Sales Order object is available to support the Oracle Sales Cloud integration with Oracle Configure, Price, and Quote (CPQ) Cloud. This object stores quote summary information that is synced from Oracle CPQ Cloud.

**Note:** Do not extend the Sales Order object unless the integration with Oracle CPQ Cloud has already been set up. For more information, see Related Files for Oracle Sales Cloud Integration with CPQ (2015009.1) on My Oracle Support at https://support.oracle.com.

After the integration has been set up, navigate to Application Composer and use the Sales Order object, to modify these pages:

- Quotes and Orders subtab for account records
- Quotes and Orders subtab for opportunity records

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).
The custom fields you add are available for display on the Quotes and Orders subtab for account and opportunity records, as well as for use with web services.

Modify the Sales Order Object
You can extend the Sales Order object only after the integration with Oracle CPQ Cloud has been set up.

Extending Quotes and Orders Subtab
To extend the Quotes and Orders subtab, available from both the Account or Opportunity details page:

1. Navigate to the Sales Order object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click either Edit Customer Quotes and Sales Orders Summary Table or Edit Opportunity Quotes and Sales Orders Summary Table.

Related Topics
- Modifying Oracle Sales Cloud Simplified Pages: Explained
15 Extending Account, Contact, and Household

Overview

Extend objects and pages related to accounts, contacts, and households in Oracle Sales Cloud.

In this chapter, you will learn about:

- Each standard object related to accounts, contacts, and households, and how to extend those objects using Application Composer.
- How to configure Customer Center pages using Page Composer.

Extending Pages for Accounts: Explained

Using Application Composer, you can change many items for accounts pages.

Use Application Composer to modify the following pages:

- Accounts landing page
- Create Account page
- Edit Account page

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

Note: To extend the pages for accounts, use the Account object in Application Composer.

Accounts Landing Page

You can modify the Accounts landing page, also referred to as the Accounts list page. For example, you can hide standard account fields or show custom account fields. To modify the Accounts landing page:

1. Navigate to the Account object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click Edit Summary Table for Account.
Create Account Page
You can modify the Create Account page. For example, you can hide standard account fields, make a field required, add a default value, and show custom account fields. To modify the Create Account page:

1. Navigate to the Account object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group custom fields into field groups.

**Note:** Custom fields that you add to the Account object and then select for display in a page layout appear on the Create Account page, before the address set of fields.

Edit Account Page
The Edit Account page consists of multiple subtabs that display along the left side of the page. You can modify most of these subtabs, as well as hide or show subtabs. For example, you can also add custom subtabs to display records from child or related objects.

To add custom fields to certain subtabs available from the Edit Account page, you must first navigate to those subtab target objects in Application Composer to create the fields. After creating the fields on the target object, you can then navigate to the required details page layout for the Account object to add those fields to the relevant subtabs.

To modify the Edit Account page:

1. Navigate to the Account object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Account page, select the subtab you want to change first. The following subtabs are available from the Edit Account page. Use the following subtabs to configure which standard and custom fields display at runtime.

   - **Overview**
   - **Profile**
   - **Team**
     - Use the Sales Account Resource and Sales Account Territory objects to create custom fields as needed.
   - **Contacts**
     - Use the Contact object to create custom fields as needed.
   - **Assets**
     - Use the Asset object to create custom fields as needed.
   - **Opportunities**
     - Use the Opportunities subtab on the Edit Account page are not reflected on the Opportunities subtab on the Edit Contact page.
Use the Opportunity object to create custom fields as needed.

- Leads
  Use the Lead object to create custom fields as needed.

- Relationships
  Use the Relationship object to create custom fields as needed.

- Notes
  Use the Note object in Application Composer to create custom fields as needed, and to extend this subtab.
  On the Simplified Pages tab for the Note object, edit the default creation page layout or duplicate it to create a new layout.
  The Notes list page is not extensible.

- Activities
  Use the Activity object in Application Composer to extend this subtab.
  On the Simplified Pages tab for the Activity object, click **Edit Embedded Summary Table** (at the bottom of the page).

- Conversations
  Not extensible from this page.

- White Space Analysis
  Sales representatives can use the White Space Analysis report to pinpoint accounts with the greatest potential.

> **Note:** This subtab is hidden by default. To use this subtab, you must first purchase the license for it. After you display this subtab, you must also set up your Sales Predictor rules.

**Related Topics**
- Modifying Oracle Sales Cloud Simplified Pages: Explained

**Extending Pages for Contacts: Explained**

Using Application Composer, you can change many items that appear on the pages for contacts.

Use Application Composer to modify the following pages:

- Contacts landing page:
  - Contacts cards view
  - Contacts listing
- Create Contact page
- Edit Contact page
For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

>Note: To extend the pages for contacts, use the Contact object in Application Composer

**Contacts Landing Page**

The Contacts landing page consists of two different views, which your users can toggle between:

- The Contacts cards view
- The Contacts listing

You can modify both pages.

The Contact cards view displays a list of all your users' contacts, but represented visually as cards. You can modify the fields that appear in each card. For example, you can hide standard contact fields or show custom contact fields. To modify the Contact cards view:

1. Navigate to the Contact object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click *Edit Landing Page*.
4. In theConfigure Card View region, select the fields that you want to display or hide in your contact cards.
5. Optionally, select the *Enable alphabetic filtering* check box to display the alphabetic bar which runs across the top of the contact cards page.

The Contacts listing displays all your users' contacts, but represented in a table list view. You can modify the columns that appear in the table. For example, you can hide standard contact fields or show custom contact fields. To modify the Contacts listing:

1. Navigate to the Contact object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click *Edit Landing Page*.
4. In the Configure Summary Table region, select the fields that you want to display or hide in the Contacts list page.

**Create Contact Page**

You can modify the Create Contact page. For example, you can hide standard contact fields, make a field required, add a default value, and show custom contact fields. To modify the Create Contact page:

1. Navigate to the Contact object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group custom fields into field groups.
Note: Custom fields that you add to the Contact object and then select for display in a page layout appear on the Create Contact page, before the address set of fields.

Edit Contact Page

The Edit Contact page consists of multiple subtabs that display along the left side of the page. You can modify most of these subtabs. For example, you can also add custom subtabs to display records from child or related objects.

To add custom fields to certain subtabs available from the Edit Contact page, you must first navigate to those subtab target objects in Application Composer to create the fields. After creating the fields on the target object, you can then navigate to the required details page layout for the Contact object to add those fields to the relevant subtabs.

To modify the Edit Contact page:

1. Navigate to the Contact object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Contact page, select the subtab you want to change first. The following subtabs are available from the Edit Contact page. Use the following subtabs to configure which standard and custom fields display at runtime.

   - **Overview**
   - **Profile**
   - **Team**
     - Use the Sales Account Resource and Sales Account Territory objects to create custom fields as needed.
   - **Assets**
   - **Opportunities**
     - Use the Opportunity object to create custom fields as needed.
   - **Leads**
     - Use the Lead object to create custom fields as needed.
   - **Relationships**
     - Use the Relationship object to create custom fields as needed.
   - **Notes**
     - Use the Note object in Application Composer to create custom fields as needed, and to extend this subtab. On the Simplified Pages tab for the Note object, edit the default creation page layout or duplicate it to create a new layout. The Notes list page is not extensible.
   - **Activities**
Use the Activity object in Application Composer to extend this subtab.

On the Pages tab for the Activity object, click **Edit Embedded Summary Table** (at the bottom of the page).

- Conversations

Not extensible from this page.

**Related Topics**

- Modifying Oracle Sales Cloud Simplified Pages: Explained

---

**Extending Pages for Households: Explained**

Using Application Composer, you can change many items that appear on the pages for households.

Use Application Composer to modify the following pages:

- Households landing page
- Create Household page
- Edit Household page

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

> **Note:** To extend the simplified set of pages for households, use the Household object in Application Composer.

**Households Landing Page**

You can modify the Households landing page, also referred to as the Households list page. For example, you can hide standard household fields or show custom household fields. To modify the Households landing page:

1. Navigate to the Household object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click **Edit Summary Table for Household**.

**Create Household Page**

You can modify the Create Household page. For example, you can hide standard household fields, make a field required, add a default value, and show custom household fields. To modify the Create Household page:

1. Navigate to the Household object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group custom fields into field groups.

Note: Custom fields that you add to the Household object and then select for display in a page layout appear on the Create Household page, before the address set of fields.

Edit Household Page

The Edit Household page consists of multiple subtabs that display along the left side of the page. You can modify most of these subtabs. For example, you can also add custom subtabs to display records from child or related objects.

To add custom fields to certain subtabs available from the Edit Household page, you must first navigate to those subtab target objects in Application Composer to create the fields. After creating the fields on the target object, you can then navigate to the required details page layout for the Household object to add those fields to the relevant subtabs.

To modify the Edit Household page:

1. Navigate to the Household object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Household page, select the subtab you want to change first. The following subtabs are available from the Edit Household page. Use the following subtabs to configure which standard and custom fields display at run time.

   - Overview
   - Profile
   - Team

   Use the Sales Account Resource and Sales Account Territory objects to create custom fields as needed.
   - Assets

   Use the Asset object to create custom fields as needed.
   - Opportunities

   Use the Opportunity object to create custom fields as needed.
   - Leads

   Use the Lead object to create custom fields as needed.
   - Relationships

   This subtab has both a Household Contacts list as well as an Other Relationships list.

   Use the Contact object to create custom fields as needed, before you configure the Household Contacts list here.
Use the Relationship object to create custom fields as needed, before you configure the Other Relationships list here.

- **Notes**
  Use the Note object in Application Composer to create custom fields as needed, and to extend this subtab. On the Simplified Pages tab for the Note object, edit the default creation page layout or duplicate it to create a new layout.
  The Notes list page is not extensible.

- **Activities**
  Use the Activity object in Application Composer to extend this subtab. On the Simplified Pages tab for the Activity object, click **Edit Embedded Summary Table** (at the bottom of the page).

- **Conversations**
  Not extensible from this page.

**Related Topics**

- Modifying Oracle Sales Cloud Simplified Pages: Explained

---

### Extending Simplified Pages for Relationships: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for relationships. Use Application Composer to modify this page:

- **Add Relationship page**

This page is available from the Relationships subtab when you drill down from the simplified details pages for these objects:

- Accounts
- Contacts
- Households

### Add Relationship Page

You can modify the Add Relationship page which displays when you create a relationship from the Relationships subtab on any Edit Account, Edit Contact, or Edit Household page. For example, you can hide standard relationship fields, make a field required, add a default value, and show user-defined relationship fields. You can also reorder fields and change field labels.

To modify the Add Relationship page:

1. Navigate to the Relationship object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
   To enhance run time usability for your end users, optionally group user-defined fields into field groups.
Extending Pages for Assets: Explained

Using Application Composer, you can change many items that appear on the pages for assets.

Use Application Composer to modify these pages:

- Asset list page (includes a summary table of assets)
- Create Asset page
- Edit Asset page

These pages are available as subtabs when you drill down from the details pages for these objects:

- Accounts
- Contacts
- Households

**Note:** Use the Asset object in Application Composer to create custom asset fields. Note that you cannot add custom buttons and actions to any page for assets.

**Assets List Page**

You can modify the Assets list page which displays when you first navigate to the Assets subtab on any Edit Account, Edit Contact, or Edit Household page. For example, you can hide standard asset fields or show custom asset fields.

To modify the Assets list page which appears on the Assets subtab on the Edit Account, Edit Contact, or Edit Household pages:

1. Navigate to the Account, Contact, or Household object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. Select the Assets subtab to configure which standard and custom fields display at run time.
   Use the Asset object to add custom fields as needed, before you can select them for display on the subtab.

**Tip:** The changes you make to the Assets subtab are unique across the Edit Account, Edit Contact, and Edit Household pages. For example, the changes you make to the Assets subtab on the Edit Account page are not reflected on the Assets subtab on the Edit Contact page.

**Create Asset Page**

You can modify the Create Asset page which displays when you create an asset from the Assets subtab on any Edit Account, Edit Contact, or Edit Household page. For example, you can hide standard asset fields, make a field required, add a default value, and show custom asset fields. You can also reorder fields and change field labels. To modify the Create Asset page:

1. Navigate to the Asset object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group custom fields into field groups.

### Edit Asset Page

You can modify the Edit Asset page which displays when you edit an asset from the Assets subtab on any Edit Account, Edit Contact, or Edit Household page. For example, you can hide standard asset fields, make a field required, add a default value, and show custom asset fields. You can also reorder fields and change field labels. To modify the Edit Asset page:

1. Navigate to the Asset object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group custom fields into field groups.

### Related Topics

- Modifying Oracle Sales Cloud Simplified Pages: Explained

### Extending Simplified Pages for Assets: Worked Example

This example demonstrates how you can modify simplified pages for assets using Application Composer.

In this example, you will see how to add the field **Asset Number** to the Create Asset page using Application Composer. You will also test whether the field appears in the simplified page for creating an asset.

> **Note:** Ensure that you have the privileges to create and view an asset.

### Extending Simplified Pages for Assets Using Application Composer

1. Navigate to Application Composer.
2. In the object tree, select the standard object, Asset, which includes a set of simplified pages.
3. Select the Pages node.
4. Select the Simplified Pages tab.
5. Use the links on the tab to navigate to the object’s configuration pages, where you can modify the simplified pages that are available for the selected object. You can show or hide fields, rearrange fields, and add your own fields.
6. For example, to add the field **Asset Number** to the Create Asset page, highlight a page layout in the Creation Page Layouts table and click the **Edit** icon. You can also select **Actions > Edit Layout**.
7. On the **Edit Simplified Creation Page**, click the Edit icon (pencil icon) to modify the Create Asset page.
8. In the Available Fields list, double-click **Asset Number** to move it to the **Selected Fields** column. You can also click the Right arrow button to move an item.
This figure shows the Available Fields and Selected Fields lists, which you use to hide and show fields on the run time Create Asset page.

9. Click Save and Close.
10. Click Done.

Viewing the Changes in the Create Asset Page

In this section, we test whether the field **Asset Number** that we added to the Create Asset page appears at run time.

1. Login to the Simplified Pages main page and click **Households**.
2. Select a household that has a customer type.
This figure shows the Household overview page with the details of the selected household.

3. Click the number in the Assets region.
4. In the Edit Household page, click the **Create Asset** button.

This figure shows the Edit Household page for the selected item. You can create an asset from this page.

**Note:** The Create Asset button is visible only if you have the Asset Creation privilege.
5. The field **Asset Number** that you added using Application Composer now appears at the bottom of the Create Asset page.

This figure shows the **Create Asset** page with the field **Asset Number** that we added as part of extending the Asset object.
Modifying the Overview Subtab: Explained

You can modify the Overview subtab for accounts, contacts, households, and partners. You can hide or show fields that appear on the left side of the subtab. Additionally, you can rearrange the position and grouping of the summary boxes, formerly referred to as infolets, that appear on the right side of the subtab. If required, you can also create new summary boxes for custom objects only.

Extensibility on the Overview Subtab

The Overview subtab for accounts, contacts, households, and partners is the top subtab that appears for an account, contact, household, or partner record. Data that appears in the Overview subtab is read only, but your users can drill down to more details from here.

On the left side of the Overview subtab, key fields for the account, contact, household, or partner record are displayed. Use Application Composer to show or hide these read-only fields.

On the right side of the subtab, key data points about related object records are displayed as number totals in summary boxes. Use Application Composer to rearrange the position and grouping of the summary boxes. You can also create new summary boxes to highlight key data points for custom objects.

Hiding and Showing Fields

The Overview subtab includes a region on the left with a standard set of fields that display. You can further modify those fields by hiding existing fields, and showing new fields. You can also reorder these fields.
**Note:** These fields are read only. Your end users can edit these fields at run time on the Profile subtab.

To hide and show fields, and reorder them, on the Overview subtab:

1. In Application Composer, navigate to the Simplified Pages tab for either the account, contact, household, or partner object.
2. On the Simplified Pages tab, scroll down to view the Details Page Layouts table.
   - Duplicate the standard layout to create a new layout to edit, or edit another existing layout.
3. Within the selected details page layout, navigate to the Overview subtab.
   - Click the Edit icon next to the Overview Form heading to configure which fields display at run time, and in which order.

---

### Summary Box

A summary box, formerly referred to as an infolet, is a container on a page which summarizes some key information about the record’s related object records, so that your end users don’t have to navigate to other tabs. For example, a summary box might display the number of total open opportunities that you’re responsible for. At run time, users can click that number to drill down to more details. Typically, clicking a number in a summary box takes the user to another subtab for the object record.

**Note:** The summary boxes that appear on the Overview subtab for accounts, contacts, households, and partners are different from the infolets that you can embed in Business Intelligence (BI) dashboards, and thus are not interchangeable.
Below is an example of three summary boxes that display on the Account Overview subtab. These three summary boxes are arranged within a summary box group called Opportunities.

**Opportunities**

![Summary Box Group](image)

The account, contact, household, and partner objects have a set number of summary boxes that you can rearrange and group. Summary boxes are grouped by default, but you can create new groups and rearrange the summary boxes to fit the needs of your users. In addition, you can create new summary boxes for custom objects only.

To view the available summary boxes for an object:

1. In Application Composer, navigate to the Simplified Pages tab for either the account, contact, household, or partner object.
2. On the Simplified Pages tab, scroll down to view the Summary Boxes table.

**Infolets**

![Infolets Table](image)

This table lists the available summary boxes that are registered for the object. Review the next section for tips on working with summary boxes and summary box groups.
Working with Summary Boxes

The Overview subtab includes a region on the right where summary boxes display. Summary boxes are grouped and arranged by default, but you can rearrange the position and grouping of these summary boxes. You can remove a summary box from an existing summary box group, and add it to another existing summary box group. Or, you can create a new summary box group, or delete existing summary box groups.

To rearrange summary boxes:

1. In Application Composer, navigate to the Simplified Pages tab for either the account, contact, household, or partner object.
2. On the Simplified Pages tab, scroll down to view the Details Page Layouts table.
   Edit any custom layout. If none exists, then duplicate the standard layout and edit the resulting custom layout.
3. Within the selected details page layout, navigate to the Overview subtab to configure summary box groups.
   Click the Edit icon next to the summary box group that you want to configure.
4. Use the Available Summary Boxes and Displayed Summary Boxes lists to control which summary boxes are displayed within this summary box group at run time. Within the Displayed Summary Boxes list, use the up or down arrows to control the order of the summary boxes within the group.
   If a summary box is listed under Available Summary Boxes but you cannot select it, then this means that the summary box is already displayed inside another summary box group. If you want to include that summary box in
the current group, then you must remove it from the other summary box group first, before you can add it to the current group.

**Details Layout: Default Layout: Edit Infolet Group:**

![Infolet Group Properties](image)

**Available Infolets**
- Open Activities
- Top Recommendations
- Assets

**Displayed Infolets**
- Open Opportunities
- Open Quotes and Orders
- Open Leads

5. You can delete summary box groups from this page by clicking **Delete**. Deleting a summary box group doesn't delete any included summary boxes.

To create a new summary box group:

1. In Application Composer, navigate to the Simplified Pages tab for either the account, contact, household, or partner object.
2. On the Simplified Pages tab, scroll down to view the Details Page Layouts table.
   - Edit any custom layout. If none exists, then duplicate the standard layout and edit the resulting custom layout.
3. Within the selected details page layout, navigate to the Overview subtab to configure summary box groups.
4. Navigate to the bottom of the page and click **Create Summary Box Group**.

**Creating Summary Boxes**

You might want to highlight custom object data on an Overview subtab. Use Application Composer to create new summary boxes that summarize key data points about custom objects.

You cannot create summary boxes for standard objects.

To create a new summary box for a custom object:

1. In Application Composer, navigate to the Simplified Pages tab for either the account, contact, household, or partner object.
2. On the Simplified Pages tab, scroll down to view the Summary Boxes table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Activities</td>
<td>Standard</td>
</tr>
<tr>
<td>Top Recommendation</td>
<td>Standard</td>
</tr>
<tr>
<td>Open Opportunities</td>
<td>Standard</td>
</tr>
<tr>
<td>Open Quotes and Orders</td>
<td>Standard</td>
</tr>
<tr>
<td>Assets</td>
<td>Standard</td>
</tr>
<tr>
<td>Open Leads</td>
<td>Standard</td>
</tr>
</tbody>
</table>

3. Click **New**.
4. Enter the name of the summary box. The name that you enter displays to your users as the heading of the summary box.
5. Select the target object for the summary box. Select from the custom objects that are related to the current object: account, contact, household, or partner.
6. Select the mathematical operation that you want the summary box to perform on the related object’s records. For example, do you want the summary box to display a count of all records?
7. Optionally add filters to the summary box to constrain the records that are included in the results. For example, maybe you want to include only records created by a specific person.
8. Click **Save and Close**.

After creating a summary box, you must add it to all details page layouts where you want the summary box to display. Remember that you can add your new summary box to only one summary box group. Use the instructions above to add your custom summary box to a summary box group.

**Related Topics**
- Extending Simplified Pages for Partners: Explained

**Enabling Display of Multiple Addresses: Explained**

The edit pages for accounts, contacts, households, and partners automatically display a primary address region. However, you can change the view to display a table that lists multiple addresses.

**How Address Details Display**

Address details for accounts, contacts, households, and partners display on their respective Profile subtabs in edit mode.
For example, the following figure shows the Primary Address region, which displays on the Profile subtab for an account record. This is the default view of address details. Use Application Composer to display multiple addresses instead of just one primary address region.

**Edit Account: ABC Corporation... : Profile**

Considerations for Configuring Address Display Tables
Before you enable the table that displays multiple addresses, consider the following:

- In Application Composer, you must ensure that only one of the address regions is exposed at a time in the Detail pages layout. For example, expose Primary Addresses or Multiple Addresses on the page, but not both at the same time.
- The default type for Address is bill-to. You can configure the default address type using Default Address Type for Account and Default Address Type for Contacts profile options.
- An account can have one or more sell-to addresses (or address type as sell-to). Account territory assignment is based on the primary address, and the primary address can be any type, such as sell-to or bill-to.
- The default type when creating an account is Prospect. You can change the default type using the Account Type Default profile option. Similarly, you can use the Contact Type Default profile option to configure the default type when creating a contact.

Procedure for Enabling the Multiple Addresses Table
Use the following procedure to enable the table that displays multiple addresses.

1. Sign in as the sales administrator or as a setup user.
2. Ensure that you are working in an active sandbox.
3. Click Navigator > Configuration > Application Composer.
4. In the Objects navigation tree, expand Standard Objects and then expand the object whose pages you want to modify. For example, select the Account object.
5. Click the Pages node.
6. Ensure that the Simplified Pages tab is selected for either the account, contact, household, or partner object.
7. In the Details Page Layouts region, edit the relevant layout.
   If none exists, then duplicate the standard layout using the duplicate layout icon, and edit the resulting layout.
8. In the Details Layout page, click the Profile tab.
   In the Primary Address region and Multiple Address region, do one of the following:
   a. Click Hide to hide the region at runtime.
   b. Click Show to show the region at runtime.
      For example, to enable the display of multiple addresses, click Hide in the Primary Address region, and click Show in the Multiple Address region.
9. Click Save and Close, then Done.
10. Publish the sandbox.
16 Extending Leads and Campaigns

Overview

Read this chapter to learn about extending the sales lead and sales campaign objects and pages in Oracle Sales Cloud. In this chapter, you will learn:

- How to modify Oracle Sales Cloud leads pages
- How to extend the sales campaign object using Application Composer
- How to create a sales lead validation rule using Application Composer
- How to create new objects by copying existing objects using the Copy Map feature
- How to map child objects and attributes in the source object to objects and attributes in the new object

Sales Lead and Sales Campaign

Extending Pages for Leads: Explained

Using Application Composer, you can change many items that appear on the pages for sales leads. Use Application Composer to modify these pages:

- Leads landing page (Landing Page Layouts)
- Create Lead page (Creation Page Layouts)
- Edit Lead Page (Details Page Layouts)
- Lead Email notifications page (Lead Notification Content)

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add user-defined fields (all types).

Note: To extend the set of pages for leads, use the Sales Lead object in Application Composer. You must be working in an active sandbox.
Leads Landing Page
You can extend the following items in the list that appears on the Leads landing page. For example, you can hide standard fields or show user-defined fields. To modify this page:

1. Navigate to the Sales Lead object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Landing Page Layouts, duplicate the standard layout to create a new layout.

Create Lead Page
You can modify the Create Lead page. For example, you can hide standard fields, make a field required, add a default value, and show user-defined fields. To modify the Create Lead page:

1. Navigate to the Sales Lead object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

To enhance runtime usability for your end users, optionally group user-defined fields into field groups.

Edit Lead Page
The Edit Lead page consists of multiple subtabs that display along the left side of the page. You can modify most of these subtabs. You can also add user-defined subtabs, for example, to display records from child or related objects.

To add user-defined fields to certain subtabs available from the Edit Lead page, you must first navigate to those subtab target objects in Application Composer to create the fields. After creating the fields on the target object, you can then navigate back to the details page layout for the Sales Lead object to add those fields to the specific subtabs.

To modify the Edit Lead page:

1. Navigate to the Sales Lead object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Lead page, select the subtab you want to change first.

The changes you make to these subtabs are unique to the Edit Lead page. For example, you can extend the Sales Lead Products directly by adding new user-defined fields, but if you want to display those new fields, or hide existing fields for the Sales Lead Products object, you must navigate to the Sales Lead object, select Pages, and then go to the Details Page Layout, Summary subtab, Lead Details Products area.

The following subtabs are available from the Edit Lead page. Use these subtabs to configure which standard and user-defined fields display at runtime.

- Summary
- Contacts
  - Use the Sales Lead Contact object to create user-defined fields as needed.
- Qualification
  - Not extensible from this page.
- Sales Team
  - Use the Sales Lead Resources object to create user-defined fields as needed.
- **Products**
  Use the Sales Lead Products object to create user-defined fields as needed.

- **Activities**
  Use the Activity object in Application Composer to extend this subtab.
  On the Simplified Pages tab for the Activity object, scroll down to the **Embedded Activity Summary Layout** section at the bottom of the page. Click the Standard layout or duplicate it to create a new layout.

- **Responses**
  Not extensible from this page.

- **Notes**
  Use the Note object in Application Composer to create user-defined fields as needed, and to extend this subtab.
  On the Simplified Pages tab for the Note object, edit the default creation page layout or duplicate it to create a new layout.
  The Notes list page is not extensible.

- **Opportunities**
  Use the Opportunity object to create user-defined fields as needed.

- **Social**
  Not extensible from this page.

**Lead Email Notifications**

This region is applicable only for partner leads. From here, you can configure actionable email content that is sent to users when partner leads are assigned. Use the Sales Lead object in Application Composer to extend the email notifications of the partner lead UI. On the Simplified Pages tab for the Sales Lead object, scroll down to the **Lead Notification Content** section at the bottom of the page. Click the Standard layout or duplicate it to create a new layout.

**Related Topics**
- Modifying Oracle Sales Cloud Simplified Pages: Explained

**Extending Pages for Leads Using Application Composer: Worked Example**

You can extend lead pages using Application Composer. You must make your changes in a sandbox, so you can test them first.

This example demonstrates how you can extend the lead object by:

- Adding a check box to identify strategic leads
- Adding a Groovy script that checks if the size of the deal is greater than 1000 USD when a user saves a lead with this check box selected. If the deal is smaller than this amount, then users are prevented from saving the lead and receive an error message instead.
Create the Strategic Deal Check Box and Add the Groovy Script Validation Check

Follow these steps to create the **Strategic Deal** check box and add the validation check:

1. Navigate to Application Composer. Select **Sales Lead** from the **Objects** panel.
2. Under the Lead object, select the **Fields** link.
3. Click the **Create a custom field** icon and, in the Select Field Type window, select **Check box**.
4. In the Create Check box Field page, enter "Strategic Deal" as the **Display Label**. Leave the rest of the fields with their default values.
5. Click **Save and Close**.
6. Next augment this new check box with a Groovy script. When the user attempts to save a record with the check box selected, then the script checks if the deal size is greater than 1000. If the deal size is less than 1000, then the script displays an error and the lead is not saved.

   Here is the Groovy logic that you can use:

   ```java
   def retVal
   if (StrategicDeal_c == 'Y')
   {
      if (DealAmount >= 1000)
      {
         retVal = true
      }
      else
      {
         retVal = false
      }
   }
   else
   {
      retVal = true
   }
   return(retVal)
   ```

7. In the Objects panel, select **Server Scripts** under the **Sales Lead** object.
8. In the **Server Scripts Sales Lead** page, select the Validation Rules tab and under **Object Rules**, select **Action - Add** to add a new validation rule.
9. In the Create Object Validation Rule page, enter the rule name as Eval and enter a simple error message in the Error Message text region.
10. Cut and paste the script that you have written to validate the condition.
11. Now you need to add this field to the simplified pages. In Application Composer, select the **Pages** link under the Sales Lead object and then select the Simplified Pages tab.
12. In the Details Page Layout region, duplicate the standard layout to create a new layout to edit.
13. Click the Edit pencil icon and select the **Strategic Deal** field.
14. Clock **Done** and save and close Application Composer.
15. Sign into the application again and drill into a lead.
16. Check if the **Strategic Deal** check box appears.
This figure shows an example of the Edit Lead Summary page with the **Strategic Deal** check box selected.

17. Enter a value less than 1000 in the **Deal Size** field.

18. Select the **Strategic Deal** check box and click **Save and Close**.

   You should receive the error message that you entered in Step 9 because the deal size violates the validation rule.

19. Reenter a value greater than 1000 in the **Deal Size** field and save.

   You have successfully extended a simplified page and added Groovy logic.

**Related Topics**

- Extending Simplified Pages: Overview

**Creating a Sales Lead Validation Rule: Worked Example**

Lead management users can change the status of a lead to Qualified, regardless of whether the lead customer is a sales account or has a primary product associated with the lead. However, to enforce compliance with your company’s lead management business processes, you want to create business rules to control when to set a lead to a Qualified status.

This example describes how to use **Application Composer** to create a validation rule that ensures a sell-to address and a primary product exist before users change the lead’s status to Qualified.

**Creating a Validation Rule for Sell-to Address**

To create a rule that validates the existence of a sell-to addresses for a sales lead, perform the following steps:

1. Navigate to **Application Composer**.
2. Select **Sales** to filter the object search.
3. From the **Objects View**, select **Standard Objects, Sales Lead, Server Scripts**.
4. Click the **Object Functions** tab, and then click the **Add a New Object Function** icon. The **Create Object Function** page is displayed.
5. In the **Function Name** field, enter the following name without spaces: `isSellToExists`.
6. In the **Returns** field, select **Boolean**.
7. In the **Function Body** area, enter the following:
   ```java
   if(null != CustomerId){
       def partySites = newView('Address');
       def criteria = partySites.createViewCriteria();
       def criteriaRow = criteria.createRow();
       criteria.insertRow(criteriaRow);
       def criteriaItem = criteriaRow.ensureCriteriaItem('PartyId');
       criteriaItem.setValue(CustomerId)
       partySites.appendViewCriteria(criteria)
       partySites.executeQuery();
   }
   while(partySites?.hasNext()) {
       def partySite = partySites.next();
       def partySiteUses = partySite?.getAttribute('PartySiteUse');
       while (partySiteUses?.hasNext()) {
           def partySiteUse = partySiteUses.next();
           if ('SELL_TO' == partySiteUse?.getAttribute('SiteUseType')) {
               return true;
           }
       }
   }
   return false;
   ```
8. Click **Validate**.
9. Click **Save and Close**.

### Creating an Enforcement Rule for Primary Product

Next, you want to create a rule for enforcing the primary product association for the sales lead as follows:

1. Click the **Add a New Object Function** icon to go to the **Create Object Function** page.
2. In the **Function Name** field, enter the following name without spaces: `isPrimaryProductAssigned`.
3. In the **Returns** field, select **Boolean**.
4. In the **Function Body** area enter the following:
   ```java
   if((null != PrimaryInventoryOrgId && null != PrimaryInventoryItemId) || null != PrimaryProductGroupId) return true; else return false;
   ```
5. Click **Validate**.
6. Click **Save and Close** to return to the **Server Scripts Sales Lead** page.

### Creating a Validation Rule for Lead Qualification

You now want to create a validation rule that checks for the existence of a sell-to address and a primary product when a user sets the Qualified status for a sales lead.

1. Click the **Validation Rules** tab and from the **Object Rules** area.
2. Click the **Add a new validation rule** icon.
3. In the **Create Object Validation Rule** page navigate to the **Rule Name** field and enter the following rule name without spaces: `qualifyLead`
4. In the **Rule Definition** area, enter the following:
   ```java
   if(isAttributeChanged('StatusCode') &&
   getAttribute('StatusCode') == 'QUALIFIED') return (isPrimaryProductAssigned() && isSellToExists());
   else return true;
   ```
5. In the **Error Message** area, enter the following message text: *Primary Product and Sell to Address are required for qualifying a Lead.*
6. Click **Save and Close**.

This completes the task of using **Application Composer** to create validation rules for the primary product and sell-to address fields when qualifying a lead.

### Testing the Rules

To validate the rules, perform the following steps:

1. Navigate to the **Lead Qualification, Edit Lead** page.
2. From the **Actions** menu, select **Qualify**.
   
   If the **Primary Product** and **Sell to Address** fields contain no data, you should receive the following error message text:
Primary Product and Sell to Address are required for qualifying a Lead.

Assigning Account Owner as the Owner of a Lead: Worked Example

You can configure Oracle Sales Cloud to automatically set a lead owner based on the owner of the associated account. For example, an employee in your organization who is not a salesperson has created a lead with an associated account. You can use a Groovy script to automatically assign that lead to the salesperson who owns the associated account.

You are a sales administrator and your management has asked you to create a script to automate setting the account owner as the lead owner after the lead is created or assigned. You can use Groovy scripts to cater for the following scenarios:

- When a lead is created with an associated account and the assignment process is run
- When a lead is created without an account, but the account is added later and the assignment process is run

>Note: You can’t set the account owner as the lead owner for leads created by sales prediction feature or for leads created or updated using file-based data import.

Setting Account Owner as the Lead Owner After Lead is Created or Assigned

Set the account owner as the lead owner after the lead is created or assigned as follows:

1. Sign in using the sales administrator role.
2. Create a sandbox and activate it.
   For more information about sandboxes, see the Sandboxes: Explained topic.
3. Navigate to Application Composer and expand Standard Objects, then Sales Lead and then click Server Scripts.
4. Click the Triggers tab, and then click the Create icon in the Object Triggers section.
5. Select the Before Insert in Database trigger type. Enter a name such as UpdatLeadOwnerOnCreate in the Trigger Name field. Copy and paste the following content of the script to the Trigger Definition section:

   ```groovy
   if(nvl(CustomerId,null) != null)
   {
     def account = newView('OrganizationProfile');
     def criteria = account?.createViewCriteria();
     def criteriaRow = criteria?.createRow();
     criteria?.insertRow(criteriaRow);
     def criteriaItem = criteriaRow?.ensureCriteriaItem('PartyId');
     criteriaItem?.setValue(CustomerId);
     account?.appendViewCriteria(criteria);
     account.executeQuery();
     if(account.hasNext())
     {
       def accRow = account.next();
       def leadResourceIterator = nvl(MklLeadResources,null);
       if(nvl(accRow?.OwnerPartyId,null) !=null)
       {
         setAttribute('OwnerId' , accRow?.OwnerPartyId);
         def resourceId = accRow?.OwnerPartyId;
         boolean alreadyOnSalesTeam = false;
         leadResourceIterator?.reset();
         for (; leadResourceIterator?.hasNext(); ) {
           def mklLeadResourcesVORowImpl = leadResourceIterator.next();
           Long existingResourceId = mklLeadResourcesVORowImpl.getAttribute('ResourceId');
           if (resourceId.equals(existingResourceId)) {
             alreadyOnSalesTeam = true;
             break;
           }
         }
       }
     }
   }
   ```
6. Click **Save and Close**. Verify that you see the newly created trigger in the **Object Triggers** section.

7. Next, click the Create icon in the **Object Triggers** section to create another object trigger.

8. Select **Before Update in Database** trigger type. Enter a name such as **UpdatLeadOwnerOnUpdate** in the **Trigger Name** field. Copy and paste the following content of the script to the **Trigger Definition** section:

   ```java
   if(isAttributeChanged('LastAssignmentDate')){
      def account = newView('OrganizationProfile');
      def criteria = account?.createViewCriteria();
      def criteriaRow = criteria?.createRow();
      criteria?.insertRow(criteriaRow);
      def criteriaItem = criteriaRow?.ensureCriteriaItem('PartyId');
      criteriaItem?.setValue(CustomerId);
      account?.appendViewCriteria(criteria);
      account.executeQuery();
      if(account.hasNext()){
         def accRow = account.next();
         def leadResourceIterator = nvl(MklLeadResources,null);
         if(nvl(accRow?.OwnerPartyId,null) !=null) {
            boolean alreadyOnSalesTeam = false;
            leadResourceIterator?.reset();
            for (; leadResourceIterator?.hasNext(); ) {
               def mklLeadResourcesVORowImpl = leadResourceIterator.next();
               if(mklLeadResourcesVORowImpl.getAttribute('ResourceId').equals(existingResourceId)) {
                  alreadyOnSalesTeam = true;
                  break;
               } else {
            leadResourceVO.setAttribute('OwnerId',accRow?.OwnerPartyId);
            } if(!alreadyOnSalesTeam) {
      def leadResourceVO = leadResourceIterator.createRow();
      leadResourceVO.setAttribute('LeadId',LeadId);
      leadResourceVO.setAttribute('ResourceId',resourceId);
      leadResourceVO.setAttribute('PrimaryFlag','Y');
   }
   }
   } else {
   if(!alreadyOnSalesTeam) {
      def leadResourceVO = leadResourceIterator.createRow();
      leadResourceVO.setAttribute('LeadId',LeadId);
      leadResourceVO.setAttribute('ResourceId',resourceId);
      leadResourceVO.setAttribute('PrimaryFlag','Y');
   }
   }
   }
   ```

9. Click **Save and Close**. Verify that you see the newly created trigger in the **Object Triggers** section.

10. Thoroughly test the following scenarios in your sandbox environment:

    - When a lead is created with an associated account and the assignment process is run
    - When a lead is created without an account, but the account is added later and the assignment process is run

11. Publish the sandbox after successfully verifying your tests.

---

**Related Topics**

- Sandboxes: Explained
Modifying the Leads UI for Sales Prospects: Worked Example

Since leads for sales prospects represent contacts with potential buying interest, some customers want to hide the lead name from the UI. Instead, they want to set the primary contact field as required, and use the primary contact as the lead identifier.

Use application composer and object triggers to cater for the following configuration tasks:

- Set the Primary Contact field as required
- Mark the name field as hidden in both the create and edit lead pages
- Automatically populate the lead name with the primary contact value when a lead is created and updated

**Note:** Sign in using the sales administrator role and create a sandbox and activate it. For more information about sandboxes, see the Sandboxes: Explained topic.

### Set Primary Contact Field as Required

Set the primary contact field as mandatory on the leads UI as follows:

1. Navigate to Application Composer and expand **Standard Objects**, then **Sales Lead** and then click **Fields**.
2. Search for the **Primary Contact** field of type **Dynamic Choice List**.
3. From the **Edit Standard Field: Primary Contact** page, under the **Constraints** section, select the **Required** check box.
4. Click **Save and Close**.

### Hide the Name Field

Hide the Name field on the leads UI as follows:

1. Navigate to Application Composer and expand **Standard Objects**, then **Sales Lead** and then click **Pages**.
2. Navigate to **Sales Lead > Pages > Creation Page Layout > Create a Duplicate Layout** page.
3. From the **Creation Page Layout** section, select **Duplicate Layout** from the **Actions** menu.
4. In the **Duplicate Layout** dialog, enter **Default User-Defined Layout** in the **New Layout Name** field.
5. Click **Save and Edit**.
6. Click the **Edit** icon to display the **Configure Detail Form**.
7. Move **Name** from **Selected Fields** list to **Available Fields** list.
8. Click **Save and Close**, then click **Done**.
9. Repeat steps 4 to 9 to edit the **Details Page Layout** section.

### Populate the Lead Name with the Primary Contact Value

Automatically populate the lead name with the primary contact value when a lead is created and updated as follows:

1. Navigate to Application Composer and expand **Standard Objects**, then **Sales Lead** and then click **Server Scripts**.
2. Click the **Triggers** tab, and then click the **Add a new Trigger** icon in the **Object Triggers** section.
3. In the **Trigger** field, select the **Before Insert in Database** trigger type.
4. In the **Trigger Name** field, enter a name, without spaces, such as **UpdatLeadOwnerOnCreate**.
5. Copy and paste the following content of the script to the **Trigger Definition** section:

   ```java
   if(PrimaryContactName == null)
   {
      throw new oracle.jbo.ValidateException('Primary Contact is a required field. Please select a Primary Contact.')
   }
   ```
6. Click **Save and Close**. Verify that you see the newly created trigger in the **Object Triggers** section.

7. Next, click the **Add a new Trigger** icon in the **Object Triggers** section to create another object trigger.

8. In the **Trigger** field, select the **Before Update in Database** trigger type.

9. In the **Trigger Name** field, enter a name such as `UpdatLeadOwnerOnUpdate`.

10. Copy and paste the following content of the script to the **Trigger Definition** section:

    ```javascript
    if(PrimaryContactName == null)
    {
        throw new oracle.jbo.ValidationException('Primary Contact is a required field. Please select a Primary Contact.'
    }
    else if(isAttributeChanged('PrimaryContactName') && PrimaryContactName != null) {
        setAttribute('Name',PrimaryContactName)
    }
    ```

    11. Click **Save and Close**. Verify that you see the newly created trigger in the **Object Triggers** section.

    12. Thoroughly test all changes in your sandbox environment:

    13. Publish the sandbox after successfully verifying your tests.

---

### Configuring Sales Campaign Pages: Explained

Use Application Composer to create user-defined fields for the sales campaign object. You can then add the user-defined fields for display in the Sales Campaign work area. This topic describes the configurable options for sales campaigns.

To access Application Composer, select Application Composer from the Navigator menu, under the Configuration category.

#### Quick Steps

To configure the sales campaign work area, follow these steps:

1. Navigate to Application Composer, expand the Standard Objects tree node, then select Sales Campaign.
2. Select the **Fields** node to create user-defined fields, or modify standard fields.
3. Select the **Pages** node to show or hide fields, either user-defined or standard.
4. Use the links on the tab to navigate to the object’s configuration pages. From there, you can configure the pages that are available for the selected object.

   > **Note:** You must duplicate the standard layouts before you can make changes.

5. Test your work by navigating to the Sales Campaign work area to view your changes.

### Understanding Which Sales Campaign Pages Are Configurable

The number of fields that you can edit or add to this region depends on the status of the sales campaign. The panel will adjust based on how many fields you add. The name of the region changes depending on where you are in the sales campaign flow.
The following list shows which regions and pages are configurable, along with some notes for each.

<table>
<thead>
<tr>
<th>Region and Page</th>
<th>Are the Fields Editable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Campaign landing page</td>
<td>No</td>
</tr>
<tr>
<td>Campaign details region on the Create Sales Campaign: Wrap Up page</td>
<td>Yes</td>
</tr>
<tr>
<td>Campaign details region (displays on Sales Campaign Overview pages for completed campaigns)</td>
<td>No</td>
</tr>
<tr>
<td>Campaign details region (displays on Sales Campaign Overview page for campaigns that are in progress)</td>
<td>No</td>
</tr>
<tr>
<td>Campaign Summary region on the Create Sales Campaign: Enter Campaign Details page</td>
<td>Yes</td>
</tr>
<tr>
<td>Campaign Summary regions on other Sales Campaign pages</td>
<td>Only if the campaign is in Draft status. Otherwise, not editable.</td>
</tr>
<tr>
<td>Campaign Details region (displays on Sales Campaign pages for completed campaigns)</td>
<td>No</td>
</tr>
</tbody>
</table>

**Adding or Modifying Object Fields**

To add user-defined fields to the Sales Campaign regions listed, first create your user-defined fields by selecting the Fields node under the sales campaign object. Then, select either a standard field to modify, or create a user-defined field.

For standard fields, you can modify these options:

- Display Label
- Required check box
- Short Description
- Updatable
- Depends On

**Adding Your Changes to the Run Time Application**

After you create user-defined fields for the sales campaign object, use Application Composer's configuration pages to add the user-defined fields to the configurable regions. You access the configuration pages in Application Composer from the Pages node under the Sales Campaign object.

To access the Pages Overview page:

1. Navigate to the application Composer Overview page.
2. In the object tree, select the Sales Campaign object.
3. Select the Pages node.
4. On the Pages tab, select the configuration page link related to the Sales Campaign page that you want to configure.

Notes About Sales Campaign Configuration
The following configuration options are not supported for sales campaigns:

- Import and export (you can’t import sales campaigns)
- Object workflow
- E-mail templates
- Business processes

The following configuration options do not apply to sales campaigns:

- Role security
- Subject areas.

Related Topics
- Defining Pages: Explained
- Defining Fields: Explained
- Sales Campaigns: Overview

Copy Map

Creating a Copy Map: Procedure
Copy maps enable you to create new objects by copying existing objects. For example, you can copy responses to create new leads and opportunities, or you can copy deal registrations to create new opportunities. A mapping file is also available to copy opportunities. You use the copy maps feature to map child objects and attributes in the source object (From object) to objects and attributes in the object you are creating (To object).

Predefined mapping files are available to you as part of your sales application to enable you to:

- Map objects and attributes during the creation of an account when converting a lead to an opportunity.
- Automatically add the primary partner contact to the lead team when a partner contact is added to the lead.
- Create leads and opportunities from responses
- Copy opportunities and create opportunities from deal registrations

To create a new copy map:

1. Sign in as a user with Application Composer access and verify that you have an active sandbox.
2. Access Application Composer by selecting Application Composer from the Navigator menu, under the Configuration category.
3. Select Sales to filter the object search, and under Advanced Setup, select Copy Maps.
4. Click New, and then enter a unique name for the copy map. You specify the unique name in a profile option after you have completed and saved your copy mapping file.
The Edit Copy Map window appears and you use it to define the copy map.

5. In the **Application Module Mapping** section, enter general information about the mapping as outlined in the following fields in the table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the mapping file. The name is displayed in the Copy Maps window. You can use the name to locate and search for copy maps that you want to edit or delete.</td>
</tr>
<tr>
<td>To</td>
<td>Select the application module that you’re copying to. For example, if you’re copying leads to opportunities, then select the application module that includes leads.</td>
</tr>
<tr>
<td>From</td>
<td>Select the application module that you’re copying from. For example, if you’re copying responses to leads, then select the application module that includes responses.</td>
</tr>
</tbody>
</table>

6. In the **Entity Mappings** section, add a record for each view object that you’re copying by completing the following fields in the table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Select the new view object that you’re copying to.</td>
</tr>
<tr>
<td>From</td>
<td>Select the existing view object that you’re copying from.</td>
</tr>
<tr>
<td>Query Type</td>
<td>Select one of the three query types: Unique Identifier, Join, or All Records. If you select All Records, then don’t enter any information in the remaining fields.</td>
</tr>
<tr>
<td>Query Attribute</td>
<td>If you selected either Unique Identifier or Join in the Query Type field, then select an attribute to use during runtime. The attribute is used to filter the records that are copied to the new object. For example, if you select LeadId, then during runtime, only the records matching the Lead ID passed to the copy map engine are copied.</td>
</tr>
<tr>
<td>Joined View Object</td>
<td>If you selected Join in the Query Type field, then use these two fields to specify the following:</td>
</tr>
<tr>
<td>and Joined</td>
<td>- The view object to which this object is joined</td>
</tr>
<tr>
<td>Attribute</td>
<td>- The attribute used as the basis of the join</td>
</tr>
</tbody>
</table>

7. Select each line in the **Entity Mappings** section, and add records to the **Attribute Mappings** section for each of the attributes by completing the following fields in the table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Select the attribute to which the selected entity is copied.</td>
</tr>
<tr>
<td>Referenced View Object</td>
<td>If this attribute is a foreign key, select the view object joined by this foreign key. The application generates new foreign keys that keep the reference intact.</td>
</tr>
<tr>
<td>Primary Key</td>
<td>If this attribute is a primary key, select this check box. Instead of copying the value in the From object, the application generates a unique value for this key field in each record in the To object.</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Select the attribute from which the selected entity is copied.</td>
</tr>
<tr>
<td>From Expression</td>
<td>Optionally, you can enter a Groovy expression to change the value in this attribute. For example, you want to change the value of the From object to some new value in the attribute of the To object. You can also enter a constant to fill this attribute with a constant value in every record of the To object.</td>
</tr>
</tbody>
</table>

8. Click **Save and Close**.

9. Specify the mapping file name in one of the following profile options:
   - Use the **Direct Lead to Opportunity Mapping** profile option to specify the mapping file name created when a direct lead is converted to an opportunity.
   - Use the **Copy Opportunity Mapping** profile option to specify the mapping file name to use when copying an opportunity. The file is used to copy attributes and child objects when creating a copy of an existing opportunity.
   - Use the **Partner Lead to Opportunity Mapping** profile option to specify the mapping file name created when a partner lead is converted to an opportunity.
   - Use the **Lead to Account Mapping** profile option to specify the mapping file name created in Copy Maps. This file maps objects and attributes during the creation of an account when converting a lead to an opportunity.
   - Use the **Lead to Contact Mapping** profile option to specify the mapping file name created in Copy Maps. This file maps objects and attributes during the creation of a contact when converting a lead to an opportunity.

---

### Modifying the Lead to Opportunity Copy Map: Worked Example

Copy maps are a declarative way to specify what fields are copied among objects. The functionality for converting a lead to an opportunity uses the predefined **Copy Lead To Opportunity Map** copy map feature. This worked examples shows you how to modify an existing mapping file to create an opportunity from a lead.

Predefined mapping files are available to you as part of your sales application. You can use copy maps to:

- Map objects and attributes during the creation of an account and the creation of a contact when converting a lead to an opportunity.
- Create leads and opportunities from responses
- Copy opportunities and create opportunities from deal registrations

In this example, you want to:

- Modify the lead to opportunity copy map
- Add a field to an opportunity
- Verify your changes

### Modifying the Lead To Opportunity Copy Map

To modify the Copy Lead to Opportunity Map:

1. Sign in as a user with Application Composer access and verify that you have an active sandbox.
2. Access Application Composer by selecting **Application Composer** from the **Navigator** menu, under the **Configuration** category.
4. Select the Copy Lead To Opportunity Map. The Edit Copy Map window appears.
5. In the Entity Mappings region, highlight the row that contains From: Sales Lead and To: Opportunity.
6. In the Attribute Mappings region, select the Add icon to add a new row.
7. From the To: drop-down list, select Comments.

This value represents the Comments field in opportunities that contains the value.
8. In the From Expression field, enter the following Groovy expression:
   `return "Copy Map executed for Lead: " + nvl(LeadNumber,"" );`

   The expression uses text and the LeadNumber as the value to set into the opportunity Comments field.
9. Click Save and Close.

Adding a Field to an Opportunity

The Comments field is not displayed by default on the Opportunities UI. Use Application Composer to add the Comments field to the opportunity header region as follows:

1. From Application Composer, expand and select Objects, Standard Objects, and then Opportunity.
2. Click Pages and then select the Simplified Pages tab.
3. On the Details Page Layouts section, edit any custom layout. If none exists, then duplicate the standard layout and edit the resulting custom layout.
4. Edit the Summary region.
5. Select the Comments field from the list in the Available Fields region and move it to the Selected Fields region.
6. Click Save and Close.
7. Verify that the Summary region list contains the Comments field.
8. Click Done.
Verifying Your Changes
To verify your changes:

1. Navigate to the Application Composer UI with the same user access and sandbox details used to complete the Modify Lead to Opportunity Copy Map task.
2. Navigate to Leads and click Create Lead.
3. Enter Autumn Lead as the Name and select Pinnacle Technologies as the Account. Accept all other defaults values for the remaining fields.
4. From the Actions drop-down list, select Convert to Opportunity.
5. Click the Opportunities icon.
6. Locate the newly created opportunity. It must have the same name as the lead that was originally created.

Examine the value in the Comments field. It contains the expected text, plus the lead number which originated this opportunity. The following figure shows an example of the Edit Opportunity page with the Comments field populated with the lead number.

Updating the Copy Map to Copy the Lead Contact Value: Worked Example

This topic describes how to update the copy map to copy over the Lead contact value into the opportunity's Customer field.

Updating the Copy Map to Copy the Lead Contact Value

In this step, you update the copy map to copy over the Lead contact value into the opportunity Customer field. Copy maps are a declarative way to specify what fields are copied among objects. The functionality for converting a lead to an opportunity uses the predefined Copy Lead To Opportunity Map copy map feature.

1. Expand Advanced Setup and click Copy Maps (in left pane of the Application Composer).
2. Click Copy Lead to Opportunity Map.
3. Under the Attribute Mappings section, in the row for TargetPartyId, select PrimaryContactId from the drop-down list in the From column.
4. Click Save and Close.
Attribute Mapping When Converting Leads to Opportunities: Explained

This topic outlines the one-to-one mapping of the attributes between the lead and opportunity applications. When converting leads to opportunities, lead attributes, such as sales account, products, revenue amount, lead contacts, and other attributes, are mapped to the newly created opportunity.

You convert a lead to an opportunity when the lead is qualified and is ready for further processing along the sales cycle. The following sections outline:

- General lead attributes mapped to opportunities
- Partner lead-specific attributes mapped to opportunities
- Lead contact attributes mapped to opportunity contact attributes
- Lead product attributes mapped to opportunity revenue line attributes
- Profile options that specify the mapping file name for user-defined mapping files

**General Lead Attributes Mapped to Opportunities**

The following table lists the general lead attributes that are mapped to the corresponding opportunity attributes:

<table>
<thead>
<tr>
<th>General Lead Attribute</th>
<th>Opportunity Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget Amount</td>
<td>Budget Amount</td>
</tr>
<tr>
<td>Budget Status</td>
<td>Budget check box</td>
</tr>
<tr>
<td>Customer</td>
<td>Target Party</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency</td>
</tr>
<tr>
<td>Description</td>
<td>Description</td>
</tr>
<tr>
<td>Estimated Close Date</td>
<td>Estimated Close Date</td>
</tr>
<tr>
<td>Expiration Date</td>
<td>Expiration Date</td>
</tr>
<tr>
<td>Lead Number</td>
<td>Lead Number</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>This attribute value is the Lead Name and the current date and time, which generates a unique opportunity name.</td>
<td></td>
</tr>
<tr>
<td>Owner</td>
<td>Owner of opportunity</td>
</tr>
<tr>
<td>Indicates the user who starts the conversion.</td>
<td></td>
</tr>
</tbody>
</table>
General Lead Attribute | Opportunity Attribute
--- | ---
Primary Contact | Key Contact
Source Code | Source

If a campaign or any other source creates a lead in which source code is used, then this attribute value is mapped.

Partner Lead Attributes Mapped to Opportunities

The following table lists the partner lead attributes that are mapped to the corresponding opportunity attributes:

<table>
<thead>
<tr>
<th>Partner Lead Attribute</th>
<th>Opportunity Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deal Approved By</td>
<td>Owner of opportunity</td>
</tr>
<tr>
<td>For direct leads, this attribute is set to the user who starts the conversion.</td>
<td></td>
</tr>
<tr>
<td>Deal Approved By Resource Organization</td>
<td>Resource Organization</td>
</tr>
<tr>
<td>Partner</td>
<td>Partner</td>
</tr>
<tr>
<td>Partner Type</td>
<td>Partner Type</td>
</tr>
<tr>
<td>Partner Program</td>
<td>Partner Program</td>
</tr>
<tr>
<td>Registration Type</td>
<td>Registration Type</td>
</tr>
<tr>
<td>Registration Number</td>
<td>Registration Number</td>
</tr>
</tbody>
</table>

Lead Contact Attributes Mapped to Opportunity Contact Attributes

To map lead contact attributes to opportunity contact attributes, the following relationships must apply between the contact and the customer:

- The relationship end date is later than the current date.
- The relationship start date is earlier than the current date.
- The relationship status is active.

The following table lists the lead contacts attributes that are mapped to the corresponding opportunity contacts attributes:

<table>
<thead>
<tr>
<th>Lead Contact Attribute</th>
<th>Opportunity Contact Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>Contact</td>
</tr>
</tbody>
</table>
Lead Contact Attribute | Opportunity Contact Attribute
---|---
Contact Role | Contact Role
Primary | Primary

**Lead Product Attributes Mapped to Opportunity Revenue Line Attributes**

The following table lists the lead product attributes that are mapped to the corresponding opportunity revenue line attributes:

<table>
<thead>
<tr>
<th>Lead Product Attribute</th>
<th>Opportunity Revenue Line Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Revenue Amount</td>
</tr>
<tr>
<td>Approver</td>
<td>Resource Party</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency for the Revenue Amount</td>
</tr>
<tr>
<td>Inventory Item</td>
<td>Inventory Item</td>
</tr>
<tr>
<td>Organization</td>
<td>Inventory Organization</td>
</tr>
<tr>
<td>Product Group</td>
<td>Product Group</td>
</tr>
<tr>
<td>Quantity</td>
<td>Quantity</td>
</tr>
<tr>
<td>Unit of Measure</td>
<td>Unit of Measure</td>
</tr>
<tr>
<td>Unit Price</td>
<td>Unit Price</td>
</tr>
</tbody>
</table>

**Profile Options for Mapping File Names**

As an administrator, you can specify your own mapping file name to use for mapping objects and attributes during the lead-to-opportunity conversion process. For example, you can create user-defined lead attributes in Application Composer, Copy Maps, and can then add these attributes to the existing ready-to-use mapping files. You must save the user-defined mapping file using a unique file name and then specify the saved mapping file name in one of the following profile options:

- Use the **Direct Lead to Opportunity Mapping** profile option to specify the mapping file name created when a direct lead is converted to an opportunity.
- Use the **Partner Lead to Opportunity Mapping** profile option to specify the mapping file name created when a partner lead is converted to an opportunity.
Modifying Lead Convert and Mass Lead Convert Pages

Use the Lead Convert and Mass Lead Convert layout pages in Application Composer if you want to expose user-defined and standard fields in your lead to opportunity conversion process. You must then configure the predefined copy map template, Copy Lead to Opportunity Map, to map the fields. Use the Manage Sales Lead Administrator Profile Values task to ensure that the configured copy map to be your default copy map which maps lead attributes to opportunity attributes during lead conversion.

Modifying the Lead Convert Pages

You duplicate the standard lead convert and mass lead convert page layouts and modify the default layouts. Edit the default page layouts to expose standard and user-defined fields so that their values can be copied to opportunity during lead conversion.

✎ Note: The following predefined standard fields exist for both the Mass Lead Convert Layouts and Lead Convert Layouts:

- Opportunity Name
- Owner
- Primary Contact
- Account

Perform the following steps to modify the lead convert pages:

1. Sign in as a user with Application Composer access and verify that you have an active sandbox.
2. Access Application Composer by selecting Application Composer from the Navigator menu, under the Configuration category.
3. From Application Composer, expand and select Objects, Standard Objects, and then Sales Leads.
4. Click Pages and then select the Simplified Pages tab.
5. On the Mass Lead Convert Layouts and Lead Convert Layouts sections, edit any user-defined layout. If no user-defined layouts exist, then duplicate the standard layouts and edit the resulting user-defined layouts, as required.
6. Click Save and Close.
7. Verify your user-defined field changes and click Done.
8. Edit the copy map to map the user-defined fields between the lead and opportunity.
   
   See the topic Modifying the Lead to Opportunity Copy Map: Worked Example for more information.
9. Set the modified copy map as your default copy map.
   
   See the topic Enabling a User-Defined Copy Lead To Opportunity Map Value in the Implementing Sales guide.

Related Topics

- Enabling a User-Defined Copy Lead To Opportunity Map Value
17 Extending Outlook

Overview

This document explains how you use Application Composer to modify the Outlook pages supported by certain standard objects.

In this chapter, you will learn:

• Which Outlook pages are extensible, and how you can modify them using Application Composer
• How to edit list views
• How to create a new form section within a standard object
• How to use synchronization settings and set up a server side filter
• How to edit scripts
• How to publish top-level and child user-defined objects
• What the different Outlook setup options are

Configuring Outlook Pages Using Application Composer: Explained

Oracle Sales Cloud provides a set of Outlook user interfaces for standard objects, such as Opportunities and Leads.

Extensible Objects

The following objects support Outlook extensibility:

• Opportunity
• Opportunity Revenue
• Sales Lead
• Account
• Contact
• Activity

Tasks on Application Composer

You can do the following tasks using Application Composer:

• Make configurations that are role-based.
• Edit list views.
• Edit and extend form sections.
• Synchronize filter settings.
• Edit scripts.
• Configure first run assistant settings.
• Configure log forwarding settings.

Configuring Outlook Pages Using Application Composer: Procedure

To configure Outlook pages using Application Composer:

1. Sign in as a sales administrator or as a setup user.
2. Ensure that you are working in an active sandbox.
3. Click Navigator > Configuration > Application Composer.
4. In the navigation tree, expand Standard Objects, and then expand the object that offers a set of Outlook pages, such as Opportunity.
5. Within the object that you're modifying, click the Pages node.
6. Select the Outlook Pages tab.
   The Outlook Pages links appear.
7. Use the options on Outlook Pages, such as Form Regions, to navigate to the object’s configuration pages, where you can configure the Outlook pages that are available for the selected object.

8. Click Save and Close, and then click Done.

9. Publish the sandbox.

Editing List Views: Procedure

You can use list views in Application Composer to define how you view objects in Outlook. In your Outlook, click Home and then click Change View to select the required view. For example, if you want to look at all opportunities, you can navigate to Change View and select All Opportunities.

Using Application Composer, you can edit the columns displayed in each view in Outlook. In this example, you add the columns Sales Stage and City to the All Opportunities view.

To edit list views:

1. Click Edit List Views in List Views on the Outlook Pages tab.

The List Views page appears.

2. Select the list view that you want to edit in the List Views region, and click the Edit icon. For example, select All Opportunities.

3. Select the desired fields and move them to the Selected Fields region. For example, move Sales Stage and City to the Selected Fields region.

   Note: You can select multiple fields at once and move them to Selected Fields. You can double-click fields to select instead of using arrows. You can move selected fields up or down to change the order of the columns in the table.

4. Click Save and Close.

5. Click Done.

6. Synchronize all changes on Outlook. See the Synchronizing and Validating Changes: Explained topic for details.

   The selected columns will now appear in the All Opportunities view in Outlook.

Configuring Outlook Forms: Explained

You can use Application Composer Outlook pages to edit the fields that are displayed in the forms on your Outlook. You can edit, add, and remove sections and fields.

Note: Not all fields that are available for configuration in the Simplified UI are available in Outlook.

You can choose to display the object that you are working on, for example Opportunity, to appear as a folder on Outlook by selecting the Show as Folder option on the edit form pages.

Use the following regions to configure forms in your Outlook:

- Object Access: Enable or disable access. You can enable or disable the create, update, and delete operations for the object on your Outlook client.
• Form Regions: Edit existing form fields and add a new form section.
• Child Regions: Edit a specific child region and add a new child region.

Editing Form Fields: Procedure

This topic provides an example of editing the header section of the Opportunity form to remove and add fields to the form.

1. In the **Opportunity Form** section under **Form Regions** on the **Outlook Pages** tab, select **Standard layout**.
2. Create a copy of the standard layout and click the **Edit Layout** icon.

   The Configure Form page appears.

3. Click the **Edit** icon on the column you want to edit. For example, click the **Edit** icon on the first column.

   The Edit Opportunity Form page appears and contains the Configure Form section.

4. Remove the fields **Strategic Value** and **Win Probability** by moving them to **Available Fields**.

   **Note:** You cannot move fields that are displayed out-of-the-box to different sections or regions.

5. Select the fields **Decision Level** and **Country** by moving them to **Selected Fields**.
6. Click **Save and Close**.
7. Similarly, select and add fields **Close Date** and **Revenue** in the second column under **Form Regions**.
8. Synchronize all changes on Outlook. See the **Synchronizing and Validating Changes: Explained** topic for details.
9. Verify your changes in Outlook.

Creating a Form Section: Procedure

This topic provides an example of adding a form section **Opportunity Details** to the Opportunities object.

1. In the **Opportunity Form** section under **Form Regions** on the **Outlook Pages** tab, select **Standard layout**.
2. Create a duplicate layout
3. Click the **Edit Layout** icon of the layout you have created.

   The Edit Opportunity Form page appears.

4. Click **Add New Section**.

   The Add New Form Section page appears.

5. Enter the **Section Name**. For example, enter Opportunity Details.

   **Note:** The section name that you enter is only for internal reference. It will not appear on the UI.

6. Select the number of columns you want in the new section from the **Selected Layouts** options. For example, select **Two column**.
7. Click **Save and Close**.
The new section is added to the Edit Opportunity Form page.

8. Click the **Edit** icon on the first column.

The Edit Opportunity Details Column 1 page appears.

9. Select the **Show Header on Form** check box.

**Note:** Select this only if you want a label to appear as a header for your new section.

10. Enter the **Region Name**. For example, enter Opportunity Details.

11. Select and add the desired fields in the Configure Form region. For example, select the fields **Consumer First Name**, **Consumer Last Name**, **Billing Account**, and **Comments**.

12. Click **Save and Close**.

The Edit Opportunity Form page appears with the new form section added.

13. Similarly, select and add fields **Country**, **City**, and **Created By** for column 2 of the new region.

14. Synchronize all changes in Outlook. See the Synchronizing and Validating Changes: Explained topic for details.

15. Verify your changes in Outlook.

### Creating a Child Section: Procedure

You can use the Child Region on Outlook Pages to:

- Select the different child views that you want appearing in Outlook.
- Select the columns you want within each child view.
The following image highlights the child regions on Outlook form:

In this example, you add a child view for Sales Lead in the Opportunity object and then add columns to the view.

To add a new child view:

1. Click the Edit icon in the Child Regions region on the Edit Opportunity Form page.
   
   The Opportunity Child Views page appears.

2. Select Sales Lead from Column 1 by moving it to Selected Objects.

3. Click Save and Close.
   
   The Sales Lead view is added to the child views. You can now edit it to select columns that you want for this view.
The following figure displays the Sales Lead view in the child view.

![Sales Lead view in child view]

In this example, let's add the columns Business Unit and Creation Date.

To select columns for the child view you just created:

1. Click the Edit icon for Sales Lead link under Child Regions.
2. Select Business Unit and Creation Date by moving them to Selected Fields.
3. Click Save and Close.
4. Click Done.
5. After you complete your changes for form regions, synchronize all changes on Outlook. See the Synchronizing and Validating Changes: Explained topic for details.
6. Verify your changes in Outlook.

Creating a Role-based Form Layout: Procedure

You can create forms to be configured for specific user roles. In this example, you create an opportunity form layout meant only for sales representatives.

To create a form for a specific user role:

1. Select Standard layout under Opportunity Form in Form Regions.
2. Click Actions, and then click Duplicate Layout.
   
   The Duplicate Layout dialog box appears.
3. Enter the layout name, such as Opportunity Layout for Sales Representative, and click Save and Close.
   
   The new layout is listed under Form Regions.
4. Click the Role Name arrow next to Any.
   
   The Select: Roles dialog box appears.
5. Select the Specific role option.
6. Select the Sales Representative Duty role by moving it to Selected Roles.
7. Click OK.

You have now defined the role for this layout.

To edit this layout, select the layout and click the Edit icon.
Synchronization Settings: Explained

Use synchronization settings to control which fields you want to replicate in Outlook during synchronization. Synchronization settings control whether your object is editable or read only in Outlook. Select the Read Only check box if you do not want users to make changes to the object in Outlook.

Sign in to Application Composer and navigate to the Outlook Pages tab for a specified object, such as Opportunity. Access the link Edit Synchronization Settings under Data Synchronization. Use the Configure Synchronization page to configure field-level synchronization settings.

The synchronization configuration contains four settings:

- Filterable
- Change Key Field
- Natural Key
- Back Update

The following is a description of these settings:

Filterable
The filterable check box determines whether end users can use a particular field while defining client-side filters. Selecting the filterable check box for a field allows end users to use that field as part of a filter while synchronizing data with the Outlook client.

Change Key Field
The change key field check box is selected by default for fields that you add using Application Composer. Any change in one of the change key fields is considered as an update on the server record and the Outlook client downloads the entire record during the next synchronization.

Note: If there is any change in the change key field, Outlook synchronizes the entire record, not only the updated field.

Natural Key
All fields selected as natural keys are used for deduplication. When you create a new record in Outlook, the deduplication process checks the server for duplicate records. It also prevents you from creating records in the Outlook client that already match the Natural Key. For example, the Natural Key is First Name and Last Name, and you try to create a new record. If another record already exists in Outlook with the same first name and last name, you get a message that the record already exists and you cannot create it.

Back Update
Back update is a setting that determines whether a field will be updated in local storage (Outlook) by a value, which is returned by the server. If a field is configured for back update, any updates to the field on the server, happening during the same synchronization, will be propagated back to the Outlook client.
Server Synchronization Filters: Explained

You can use server synchronization filters in Application Composer to create server-side filters in addition to the default filters and filters that end users set up in the control panel on the Outlook client.

Using the control panel, end users determine what data they want to replicate to Outlook. However, an administrator can create additional filters here to determine the data that is synchronized. Server-side filters are enforced first, and then end user filters on the control panel filter the data further.

Server Synchronization Filter Types

Server synchronization filters are of two types:

- Direct Filter Criteria: Use for top-level or parent objects, such as Opportunity.
- Indirect Filter Criteria: Use for child objects, such as Contacts that appears within Opportunities.

To ensure that the child object has the same synchronization filter criteria, use the Copy Direct Filter button to copy the criteria being used for the parent object.

Setting up a Server-side Filter: Procedure

You can set up filters on the server to control what data you want to synchronize to Outlook. In this example, you add a filter to ensure that only opportunities that closed in the last six months from the current date are synchronized to Outlook.

1. Navigate to the Outlook Pages tab for the Opportunity Object.
2. Select Standard layout under Edit Server Synchronization Filters in the Data Synchronization region.
3. Click Edit Server Synchronization Filters under Data Synchronization.

Note: If you want to set up a common filter for all users, use the standard layout. To set up filters for specific user roles, see the topic Creating Role-based Synchronization Filter.

4. Click the Edit Filter icon. The Configure Synchronization Filters page appears.
5. The following table describes the fields in the Direct Filter Criteria region.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Close Date</td>
</tr>
<tr>
<td>Operator</td>
<td>Between</td>
</tr>
<tr>
<td>Value</td>
<td>Literal Value</td>
</tr>
<tr>
<td></td>
<td>Current date and the date before six months.</td>
</tr>
<tr>
<td></td>
<td>For example: 10/16/14 and 4/16/14.</td>
</tr>
</tbody>
</table>
Note: Percentage field values must be between 0 and 1.

This filter ensures that only those opportunities that closed in the last six months are synchronized to Outlook.

You must now set the same criteria for the child objects.

6. Click Copy Direct Filter.

The criteria is copied to the Indirect Filter Criteria region and applied to the child objects.

Similarly, you can apply other filters on the server-side for any selected object.

Creating a Role-based Synchronization Filter: Procedure

You can create synchronization filters to be available only for specific user roles. In this example, you create an opportunity filter layout meant only for a sales representative.

To create a filter for a specific user role:

1. Select Standard layout under Edit Server Synchronization Filters in the Data Synchronization region.
2. Click Actions, and then click Duplicate Filter.

The Duplicate Filter dialog box appears.

3. Enter the new filter name, such as Sales Representative Filter, and click Save and Close.

The new filter is listed under Edit Server Synchronization Filters.

4. Click the Role Name arrow next to Any.

The Select: Roles dialog box appears.

5. Select the Specific role option.
6. Select the Sales Representative Duty role by moving it to Selected Roles.
7. Click OK.

You have now defined the role for this filter.

To edit this filter, select the filter and click the Edit Filter icon.

Editing Scripts: Explained

You can use Application Composer Outlook Pages to access advanced scripting functionality in Outlook. Scripts can be configured or predefined. Configured Event Scripts are scripts that you create. Predefined Event Scripts are scripts that are available out-of-the-box.

Note: Do not update predefined event scripts without consulting Oracle Support Services.

Write scripts using JavaScript. When you add a new script, you can choose from the event script types, as described in the following table.
## Event Type | Description
--- | ---
Before Create | Called when an item is about to be created. Can be canceled if the context is not correct.

After Create | Called after an item is created, to perform user-defined post-initialization. If created items are opened on the form, they are not saved until the After Update notification is received. Items that you create without the form must be saved explicitly.

Before Update | Called when an item is about to be updated. Can be used to update data before saving to enable user-defined logic on the form. Can be canceled if any checks fail.

After Update | Called after an item is successfully updated.

After Shown | Called after an item is shown. Used to tune form representation, such as, hide/show controls, labels, and so on.

Action Requested | Action executed in context of the form.

Sign in to Application Composer and navigate to the Outlook Pages tab for a specified object, such as Opportunity. Access the link Edit Scripts under Scripts to write your scripts.

### Creating a Script: Procedure

This topic illustrates creating scripts for two scenarios. You create scripts to ensure the following:

- Default the Purchase Date to Today for the **Asset Information** Object (Account Child Object).
- Require Address Line 1 to be populated if an address is partially filled for the **Contact** Object.

To write scripts:

1. Sign in as the sales administrator or as a setup user.
2. Ensure that you're working in an active sandbox.
3. Click **Navigator > Configuration > Application Composer**.
4. In the navigation tree, expand **Standard Objects**, and then expand the object you want to modify. For example, expand the **Accounts** object.
5. Within the object you're modifying, click the **Pages** node.
6. Click the **Outlook Pages** tab.
7. Click **Edit Scripts** under **Scripts**.
   The List Views page appears.
8. Click the **Create Event Script** icon.
   The Edit Event Script page appears.
9. Select **After Shown** from the **Event Type** list.
10. Write the following script in the Script Content field:
    
        ```javascript
        var item = get_item()
        if (item.is_new())
        {
        ```
```
var date = new Date();
var field = "PurchaseDate"
            date.setDate(date.getDate())
      item.set_property(field, local_to_utc(date.getVarDate()))
}

11. Click **Save and Close**.

   This script ensures that the purchase date defaults to the current date for the Asset Information object. Similarly, you create a script for the address line requirement for the Contact object.

   1. Click **Pages** under the **Contact** standard object.
   2. Click **Edit Scripts**, and then click the **Create Event Script** icon to navigate to the Edit Event Script page.
   3. Select **Before Update** from the **Event Type** list.
   4. Write the following script in the **Script Content** field:

```
var item = get_form().get_item();
var address_h = new customization_helpers.contact_address_handler(item);
if (!address_h._is_address_empty() && customization_helpers.is_null_or_empty(item.get_property("Address Line 1")))
|
message_box
{
"You must enter Address Line 1",
res_string("msg_validation_failed")
);
cancel()
}

5. Click **Save and Close**.

   This script ensures that the address line 1 is filled if an address is partially filled for the Contact object.
```

### Configuring User-Defined Objects for Outlook Pages: Explained

This chapter details steps to add user-defined objects to Outlook Pages.

To add user-defined objects to Outlook Pages:

1. Create your user-defined objects and publish them.

   ✍️ **Note:** User-defined objects are available in web services only when they are published. Therefore, they must be published so that they are available and can be used for Outlook changes.

   For information on creating and publishing user-defined objects, see the sections **Defining Objects: Explained** and **Publishing Sandboxes: Procedures** in the Oracle Sales Cloud Extending Sales guide.

2. Start a new sandbox and add the user-defined object to Outlook.
3. Test your application changes.
4. Publish this sandbox.

No Outlook pages are defined for user-defined objects by default. So, after you create a user-defined object, navigate to **Pages**, click **Outlook Pages**, and then click **Add to Outlook** to configure your object.
For some objects such as Customer Center, you must explicitly assign security permission to the sales representative so that synchronization completes successfully.

After you configure the object, the Outlook Pages tab displays the same links as for standard objects, as displayed in the following figure.

![Field Mapping: Pages](image)

**Related Topics**

- Defining Objects: Explained
- Publishing Sandboxes: Procedure

### Creating Outlook Pages for Parent User-Defined Objects: Procedure

In this example, you create Outlook Pages for the parent user-defined object called **Trouble Ticket** so that the Trouble Ticket object is available as a folder in Outlook.

To create Outlook pages for the user-defined object Trouble Ticket:

1. Sign in as the sales administrator or as a setup user.
2. Ensure that you are working in an active sandbox.
3. Click **Navigator > Configuration > Application Composer**.
4. Navigate to the user-defined object **Trouble Ticket**.
5. Click **Pages**, and then click **Outlook Pages**.
6. Click **Add to Outlook**.
7. Ensure that the **Top-level Object** check box is selected.
   - Top-level object indicates that the object is a parent object that appears as a folder in Outlook.
8. Select the change key and natural key fields for the object. You can retain the default selection.
9. Click **Save and Close**.
The Trouble Ticket: Pages page appears and the Outlook Pages tab contains links to edit list views, forms, synchronization settings, scripts, and child objects.

10. Publish the sandbox.

**Editing a Parent User-Defined Object List View: Procedure**

This topic describes how to edit parent user-defined object list view.

1. Click **Edit List Views** under **List Views**, to edit the Trouble Ticket list view.

   The Edit Configure View page appears.

2. Select the desired fields by moving them to **Selected Fields**. For example, select **Abstract**, **Description**, **Type**, and **Status**.

3. Click **Save and Close**.

**Editing Parent User-Defined Object Form: Procedure**

This topic describes how to edit a parent user-defined object form.

1. Click **Edit Trouble Ticket Form** under **Form Regions**.

   The Edit Trouble Ticket Form page appears.

2. Click the **Edit** icon on the first column under **Form Regions**, and select and add the fields **Abstract**, **Description**, and **Customer**.

3. Similarly, select and add the fields **Type** and **Status** in the second column under **Form Regions**.

4. Click **Save and Close**.

5. Click the edit buttons for the first and second columns under Child Regions, and add the views **Notes** and **Appointment** respectively.

6. Click **Save and Close**.

   The Edit Trouble Ticket Form page appears.
The following figure displays how your form appears on Outlook after you synchronize your changes.

Adding User-Defined Child Object: Procedure

This topic provides an example of adding a user-defined child object on Outlook.

Note: Child objects are not supported for Activity.

In this example, you add a user-defined child object called Asset Information under the Account object in Outlook.

1. Sign in as the sales administrator or as a setup user.
2. Ensure that you are working in an active sandbox.
3. Click Navigator > Configuration > Application Composer.
4. Ensure that Common is selected from the Applications list.
5. In the navigation tree, expand Standard Objects, and then expand the object you want to modify. For example, expand the Account object.
6. Within the object you’re modifying, click the Pages node and Outlook Pages.
All links under **Outlook Pages** appear.

7. Click **Edit Asset Information** under **Child Objects**.

The Edit Asset Information page appears.

8. Click **Add to Outlook**.
The Add to Outlook page appears.

9. Select the change key and natural key fields for the object. You can retain the default selection.
10. Click **Save and Close**.
The Edit Asset Information page containing all the links appears.

11. Click **Save and Close**, then click **Done**.
12. Publish the sandbox.

### Editing Child User-Defined Object Form: Procedure

This topic describes how to edit a child user-defined object.

1. Click **Asset Information Form** under **Form Regions**.

   The Edit Asset Information Form page appears.

2. Select the fields **Asset Name** and **Purchase Date** in the first column of the form.
3. Click **Save and Close**.

   The Edit Asset Information Form page appears with the names of the fields.

4. Click **Save and Close**.
Adding Child Object to the Parent Object: Procedure

After creating and adding fields to the child object form, you must add this child object to the main form region or the parent object.

This topic provides an example of adding a child object to the parent object.

🔗 **Note**: Child objects are not supported for Activity.

1. Navigate to **Outlook Pages** for **Account** under **Standard Objects**.

2. Click **Edit Organization Form** under **Form Regions**.
The Edit Organization Form page appears.

3. Click the **Edit** icon for **Organization Child Views** under **Child Regions** to add the **Asset Information** object.
   
The Organization Child Views page appears.

4. Select **Asset Information** under Column 2.
5. Click **Save and Close**.
Asset Information appears under Child Regions.

This figure illustrates how the Asset Information child region appears in Outlook after you synchronize changes.

Defining Many-To-Many Relationships for Child Objects:

Procedure

You can define many-to-many relationships in Outlook between a source object and a target object using Application Composer. If a many-to-many relationship exists between a source object and a target object, then an instance of the source object can be associated with multiple instances of the target object or the other way round. For example, a contact can be associated with multiple opportunities, and an opportunity can be associated with multiple contacts.
Setting up many-to-many relationship in Outlook involves the following steps:

1. Enable many-to-many relationship between the source object and the target object. This step creates an intersection object.
2. Navigate to Outlook Pages on the source or the target object, and add the newly created intersection object to Outlook.
3. Add the target (or the source) object as a child object to the source (or the target) object.

In this example, you set up a many-to-many relationship between the user-defined source object Service Request and the target object Contact.

Prerequisites

Create a user-defined object called Service Request. See the Adding Parent user-defined Object for Outlook Pages: Procedure topic for details.

Enabling a Many-to-Many Relationship

To enable a many-to-many relationship between Service Request and Contact:

1. Navigate to Application Composer and click Relationships under Common Setup.
2. The Relationships page displays all the existing one-to-many and many-to-many relationships that are already created, as displayed in the following figure.

![Relationships Page](image)

3. Click the Create icon.
   The Create Relationship page appears.
4. Select Service Request from the Source Object list.
5. Select Contact from the Target Object list.
6. Select M:M from the Cardinality list. Cardinality refers to the nature of the relationship, whether it’s one-to-many or many-to-many.

   When you select M:M, the following fields appear:
   - Intersection Object
   - Data Filter - Target
   - Data Filter - Source
7. Enter ServiceRequestContactRel in the Name field.
8. Enter ServiceRequestContact in the Intersection Object field.

   The intersection object contains the primary identifiers for related records from both source and target objects.
9. Click Save and Close.
The Relationships page appears and the many-to-many relationship that you created is displayed in the list of relationships.

Adding the Intersection Object to Outlook
For the intersection object to be available to be added to the source object, it has to be added to Outlook.

1. Navigate to Service Request under Custom Objects and click Outlook Pages.
2. Click Edit ServiceRequestContact under Child Objects.
3. Click Add to Outlook.

> **Note:** Don't make any changes to the intersection object Outlook pages.

4. Click Save and Close.
5. Click Done.

Adding a Child Object
You must add your target object Contact as the child object to the parent source object Service Request.

See the Adding Child Object to the Parent Object topic for details.

To add child objects:

1. Edit Default custom layout under Form Regions.

2. Click the Edit icon in the Child Regions section.

3. Select Contacts [ServiceRequestContacts] by moving it to Selected Objects either in Column 1 or 2.
4. Click Save and Close.

Contacts [ServiceRequestContacts] appears in the Child Regions.

5. Click the Edit icon next to Contacts [ServiceRequestContacts] to add the required fields for this region.
6. Select First Name and Last Name by moving it to Selected Fields.

> **Note:** All fields from the intersection objects will have the intersection object name appearing within parentheses. For example, Created By (ServiceRequestContacts).

7. Click Save and Close.
8. Click Done.
9. Synchronize all changes on Outlook.

See the Synchronizing and Validating Changes topic for details.

> **Note:** User-defined dynamic choice-list to products and product groups is not supported.
Selecting Outlook Setup Options: Explained

You can use Outlook Setup in Application Composer to perform various setup and administration tasks.

You can perform the following setups:

- Use Outlook Setup option to generate or extract client configuration onto the Outlook client.
- Use Client Events and Synchronization Report to manage log forwarding settings.
- Use First Run Assistant to manage first run configuration settings.

Outlook Setup

While using Outlook Setup options, consider whether you have made configuration within or outside of the Outlook flow. For configuration not specific to Outlook, for example, the default value updated for a field, or for language-specific label changes, use the Generate button to replicate this change on Outlook.

If you have made Outlook-specific configuration within Outlook Pages, use the Extract button to apply changes to Outlook. Extract exports a complete Outlook package that you can use for testing or deployment. See Extending Outlook Pages: Testing Guidelines for details.

Note: You can use the Extract button from any application within Application Composer, irrespective of which application you made the field-level changes in. The extract feature is not application-dependent.
Client Events and Synchronization Report

Use the Client Events and Synchronization Report region to configure the log forwarding settings.

Set the following synchronization report parameters:

- **Report Generation Level**: Defines the level of information you want to include in your synchronization report. Select from the following list items:

  - **Report Generation Only**: Report is generated but stored only on the user's computer.
  - **Report Generation and Upload to Network**: Report is generated and sent to the specified shared drive.
  - **Report Generation and Error Log Files Uploaded to Network Folder**: Report is generated and sent to the specified shared drive along with all the logs related to errors.
  - **Report Generation and All Log Files Uploaded to Network Folder**: Report is generated and sent to the specified shared drive along with the complete set of all the Outlook logs (whether or not they are related to an error).
  - **Disabled**: No report is generated.

- **Network Folder for Report Upload**: Enter the folder link within your organization file system where the synchronization reports are saved.

  ✍️ **Note**: You cannot secure the folder using login credentials.

- **Report Upload Time**: Specify the time when the synchronization report is run and saved. The time is in the CRON format which consists of five parts;

  - #1: minute [0 to 59]
  - #2: hour [0 to 23]
  - #3: day of the month [1 to 31]
  - #4: month [1 to 12]
  - #5: day of the week [0 to 6]

  Parts 1, 2, and 5 are supported and parts 3 and 4 remain as * characters.

  The default value is 0 9 * * 1-5, indicating that the report file is generated and sent at 9 am, each work week day, that is from Monday to Friday.
First Run Assistant
You can configure how the filter and synchronization settings are set up the first time a user runs the Outlook client.

The First Run Assistant includes the following settings:

- **Suppress Filter Settings**: Suppresses the Synchronization Control Panel Filters from displaying during the First Run Assistant.

- **Synchronization Settings**: Contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppress Synchronization Schedule</td>
<td>Suppresses the synchronization options from displaying during the First Run Assistant.</td>
</tr>
<tr>
<td>Full Synchronization Default Interval</td>
<td>Defines how often Synchronize All Changes will be run automatically (Once an Hour, Twice a Day, Daily, or Once a Week).</td>
</tr>
<tr>
<td>Local Synchronization Default Interval</td>
<td>Defines how often Synchronize Local Changes will be run automatically (Every 10 Minutes, Every 20 Minutes, Half an Hour, or Once a Day).</td>
</tr>
</tbody>
</table>

- **Advanced Settings**: Contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppress Advanced Settings</td>
<td>Suppresses the Advanced Settings from displaying during the First Run Assistant.</td>
</tr>
<tr>
<td>Automatically Share Contacts</td>
<td>Controls the default setting of sharing of Contacts.</td>
</tr>
<tr>
<td>Automatically Share Appointments</td>
<td>Controls the default setting of sharing of Appointments.</td>
</tr>
</tbody>
</table>
Synchronizing and Validating Changes: Explained

After you complete all your configurations in Application Composer, you must synchronize your changes to apply them to Outlook.

Perform the following steps:

1. Navigate to Application Composer as a sales administrator.
   To configure Outlook on Application Composer and validate your changes on Outlook, you must assign yourself the role of a sales representative or a sales manager, depending on the changes you want to test. To grant roles, access the task Manage Job Roles on Oracle Identity Management.

2. Create a sandbox and set it to active. For more information about creating sandbox, see the topic Creating a Sandbox: Example.

3. Configure your Outlook pages to suit your requirement.

4. Sign in to Outlook as the same user.

   **Note:** The user must be associated with an active Outlook deployment package.

5. Synchronize all changes to pick up the sandbox changes in the package.

6. Test and verify the changes in Outlook.

7. Once you are sure that all your changes are accurate, publish your sandbox.
   For more information on publishing sandboxes, see the topic Publishing Sandboxes: Procedures.

Managing Conflicts and Duplicates During Synchronization: Worked Example

This example demonstrates how to manage conflicts and duplicates that you may experience during synchronization between Oracle Sales Cloud and Outlook.

The following table summarizes key decisions for a scenario where you manage conflicts and duplicates during synchronization.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>When do you want to synchronize?</td>
<td>As soon as the updates are complete.</td>
</tr>
<tr>
<td>Which synchronization option should you run?</td>
<td>Synchronize All Changes</td>
</tr>
</tbody>
</table>
Task Summary

This worked example includes details of the following tasks that you perform to resolve conflicts and duplicates:

1. Creating a contact and updating an opportunity.
2. Resolving the conflict.
3. Resolving the duplicate
4. Synchronizing your changes.
5. Verifying your changes.

Creating a Contact and Updating an Opportunity

To create a contact and update an opportunity:

1. Create a contact called Mark Foley.
2. Update the sales stage on the Pinnacle Server Upgrade opportunity to Negotiation.
   You have just had a conversation with your customer and you know there is a high probability that this opportunity will close.
3. Synchronize all changes to update your changes on Oracle Sales Cloud.
The following figure shows the Synchronization Control Panel that displays a conflict and a duplicate.

Resolving the Conflict

The sales stage on the opportunity was also updated on Oracle Sales Cloud since the last synchronization. Although a salesperson has updated the sales stage to Presentation on Oracle Sales Cloud, you would like to go with your update as it is based on your recent conversation with the customer.

1. Select the **Outlook** check box on the Synchronization Control Panel to retain your changes.
2. Click **Synchronize Now**. Optionally, you can synchronize after you resolve the duplicate.
Resolving the Duplicate

You can resolve a duplicate contact. When you save a contact and the contact already exists on Oracle Sales Cloud, then you can either retain the Oracle Sales Cloud contact or create another contact.

Although all fields are identical, the Oracle Sales Cloud record also has the amount selected. Therefore, you decide to retain the Oracle Sales Cloud record. To resolve the duplicate contact:

1. Select the Sales Cloud check box on the Synchronization Control Panel to retain the updates made on Oracle Sales Cloud.
2. Click Synchronize Now to synchronize and resolve the conflict and duplicate.

Verifying Your Changes

To verify whether your changes are correct:

1. Open the contact Mark Foley on Outlook. The account name is now selected for this record. You have now verified that the Oracle Sales Cloud record is synchronized with Outlook.
2. Navigate to the Pinnacle Server Upgrade opportunity on Oracle Sales Cloud. The sales stage is Negotiation. You have now verified that the Microsoft Outlook record is synchronized with Oracle Sales Cloud.
Related Topics

• Managing Conflicts and Duplicates During Synchronization
Chapter 18
Extending Partner Relationship Management

Overview

Read this chapter to learn about configuring partner management pages in Oracle Sales Cloud. In this chapter, you will learn how to use Page Composer to:

- Configure partner pages used by your employees
- Configure external-facing pages used by your partners

Extending Simplified Pages for Partners: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for partners. Use Application Composer to modify these pages:

- Partners landing page
- Create Partner page
- Edit Partner page (read the following discussion of subtabs)

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

Note: To extend the simplified set of pages for partners, use the Partner object in Application Composer.

Partners Landing Page

You can modify the Partners landing page, also referred to as the Partners list page. For example, you can hide standard partner fields or show custom partner fields. To modify the Partners landing page:

1. Navigate to the Partner object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, click Edit Summary Table.
Create Partner Page

You can modify the Create Partner page. For example, you can hide standard partner fields, make a field required, add a default value, and show custom partner fields. To modify the Create Partner page:

1. Navigate to the Partner object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.

   To enhance run time usability for your end users, optionally group custom fields into field groups.

Edit Partner Page

The Edit Partner page consists of multiple subtabs that display along the left side of the page. You can modify most of these subtabs, including changing the order of the subtabs. You can also add custom subtabs to display records from child or related objects, for example.

⚠️ Note: To add custom fields to certain subtabs available from the Edit Partner page, you must first navigate to those subtab target objects in Application Composer to create the fields using the Fields node. After creating the fields on the target object, you can then navigate back to the desired details page layout for the Partner object to add those fields to the right subtabs.

To modify the Edit Partner page:

1. Navigate to the Partner object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout to edit. Or, edit another existing layout.
4. When making changes to the Edit Partner page, select the subtab you want to change first. The following subtabs are available from the Edit Partner page. Use the following subtabs to configure which standard and custom fields display at run time. You can also configure buttons and actions.

   💡 Tip: The changes you make to these subtabs are unique to the Edit Partner page. For example, the changes you make to the Opportunities subtab on the Edit Partner page are not reflected on the Opportunities subtab on the Edit Contact page.

   - Overview
   - Profile
      - In addition to selecting standard and custom fields on the profile subtab, you can configure page regions by adding field groups.
   - Contacts
      - Use the Contact object to create custom fields as needed.
   - Partner Account Team
      - Use the Partner Account Team object to create custom fields as needed.
   - Opportunities
      - Use the Opportunity object to create custom fields as needed.
- Enrollments
  Use the Program Enrollments object to create custom fields as needed.
- Notes
  The Notes subtab is not extensible, although you can change the tab name.
- Activities
  The Activities subtab is not extensible, although you can change the tab name.

Related Topics
- Modifying Oracle Sales Cloud Simplified Pages: Explained
## Extending Territory Management

### Configuring and Extending Simplified Pages for Territories: Explained

Using Application Composer, you can modify many items that appear on the simplified set of pages for sales territories.

Use Application Composer to modify these pages:

- Territories landing page
- Create Territories page
- Details page (see a discussion of subtabs in the Details page section)

For example, you can:

- Hide or show standard fields.
- Set fields as required.
- Set field default values.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

**Note:** To extend the simplified set of pages for territories, use the Sales Territory object in Application Composer.

### Territories Landing Page

You can modify the Territories landing page. For example, you can hide standard fields or show custom fields. To modify the Territories landing page:

1. Navigate to the Sales Territory object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Landing Page Layouts, duplicate the standard layout to create a new layout.

### Create Territories Page

You can modify the Create Territories page. For example, you can hide standard fields, make a field required, add a default value, and show custom fields. To modify the Create Territories page:

1. Navigate to the Sales Territory object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Creation Page Layouts, duplicate the standard layout to create a new layout.

To enhance run time usability for your end users, optionally group custom fields into field groups.
Details Page

The Details page consists of multiple subtabs that display along the side of the page. You can modify most of these subtabs. You can also add subtabs to display records from child or related objects, for example.

To add custom fields to certain subtabs available from the Details page, you must first navigate to those subtab target objects in Application Composer to create the fields. After creating the fields on the target object, you can then navigate back to the desired details page layout for the Sales Territory object to add those fields to the specific subtabs.

To modify the Details page:

1. Navigate to the Sales Territory object in Application Composer.
2. Select the Pages node.
3. On the Simplified Pages tab, under Details Page Layouts, duplicate the standard layout to create a new layout.
4. When making changes to the Details page, select the subtab you want to change first. The below subtabs are available from the Details page. Use the below subtabs to configure which standard and custom fields display at run time. You can also add a new tab.
   - Summary
   - Dimensions
     This tab is not extensible.
   - Inclusions
     This tab is not extensible.
   - Exclusions
     This tab is not extensible.
   - Team Members
     Use the Sales Territory Resource object to create custom fields as needed.
   - Child Territories
     Use the Sales Territory object to create custom fields as needed.

Configuring and Extending Oracle Sales Cloud Territory Management: Explained

You can modify a variety of regions in Oracle Sales Cloud Territory Management using Application Composer. Application Composer lets you create custom fields, actions, and links, which you then add for display in the run time partner management set of pages. You can also create custom child objects, which you can add to certain pages as subtabs or tree nodes.

This topic addresses the configurability of desktop pages only. To understand how to configure the set of simplified pages for these objects, if available, see the related topics on simplified pages.

To access Application Composer:

1. Go to the Navigator menu.
2. In the Configuration section, select Application Composer.
Understanding Which Territory Pages Are Extensible

To modify Territory pages, you need to know which pages and regions are extensible, and which options to select in Application Composer to make those changes.

The Sales objects that are associated Territory pages are:

- Sales Territory
- Sales Territory Proposal
- Sales Territory Resource

This table lists Territory pages and regions, and the related objects that you can access in Application Composer to modify those pages. For example, the Territory Team subtab on the Edit Territory Proposal page, Profile tab is extensible. To create custom fields that you can later add to the Territory Team tab, you must select the Sales Territory Resource object in Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Territory Page</th>
<th>Territory Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview page</td>
<td>Search dialog</td>
<td>Sales</td>
<td>Sales Territory</td>
</tr>
<tr>
<td>Overview page</td>
<td>Details region for a selected territory, Summary region on the Profile tab</td>
<td>Sales</td>
<td>Sales Territory</td>
</tr>
<tr>
<td>Overview page</td>
<td>Details region for a selected territory, Additional Information region on the Profile tab, Territory Team tab</td>
<td>Sales</td>
<td>Sales Territory Resource (child object of Sales Territory)</td>
</tr>
<tr>
<td>Overview page</td>
<td>Current Territory Proposals region</td>
<td>Sales</td>
<td>Sales Territory Proposal</td>
</tr>
<tr>
<td>Overview page</td>
<td>Completed Territory Proposals region</td>
<td>Sales</td>
<td>Sales Territory Proposal</td>
</tr>
<tr>
<td>Creation page</td>
<td>Create Territory Proposal dialog from the Current Territory Proposals region</td>
<td>Sales</td>
<td>Sales Territory Proposal</td>
</tr>
<tr>
<td>Details page</td>
<td>Details region for a selected territory, Profile tab, Summary region</td>
<td>Sales</td>
<td>Sales Territory Proposal</td>
</tr>
</tbody>
</table>

Next, expose your custom fields by accessing the appropriate Application Composer configuration page, listed in the following section.

*Note:* To make only minor user interface changes to Territory pages without creating new objects or fields, use Page Composer instead of Application Composer.
Adding Your Changes to the User Interface Pages

To add custom fields to the Territory regions listed in the table in the previous section:

1. Navigate to the Application composer Overview page.
2. In the object tree, select the **Sales Territory** or **Sales Territory Proposal** object.
3. Select the **Fields** node and create your custom fields.
4. Select the **Pages** node.
5. On the Desktop Pages tab, select the configuration page hyperlink related to the Territory region that you want to modify.

This table indicates which Sales objects populate which Territory pages and regions, as well as Application Composer configuration pages where you can make user interface changes on those pages and regions.

<table>
<thead>
<tr>
<th>Business Object</th>
<th>Configuration Page in Application Composer</th>
<th>Related Territories Page</th>
<th>Related Territories Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Territory</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Edit Local Search.</strong></td>
<td>Search Territories page from the Overview page</td>
<td>None</td>
</tr>
<tr>
<td>Sales Territory Proposal</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Current Territory Proposals Page.</strong></td>
<td>Manage Territory Proposals page</td>
<td>Current Territory Proposals region</td>
</tr>
<tr>
<td>Sales Territory Proposal</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Completed Territory Proposals Page.</strong></td>
<td>Manage Territory Proposals page</td>
<td>Completed Territory Proposals region</td>
</tr>
<tr>
<td>Sales Territory Proposal</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Edit Creation Page.</strong></td>
<td>Manage Territory Proposals page</td>
<td>Create Territory Proposal dialog from the Current Territory Proposals region</td>
</tr>
<tr>
<td>Sales Territory Proposal</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Edit Summary Form.</strong></td>
<td>Territory Proposal page</td>
<td>Details region for a selected territory, Profile tab, Summary region</td>
</tr>
<tr>
<td>Sales Territory Proposal</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Edit Summary Form.</strong></td>
<td>Territory Proposal page</td>
<td>Basic Information tab on the Edit Territory Proposal dialog from the Territory Proposal page</td>
</tr>
</tbody>
</table>

534
20 Extending Sales and Opportunity Management

Extending Sales and Opportunity Management Objects in Oracle Sales Cloud: Overview

Read this chapter to learn how to extend sales and opportunity management objects and pages in Oracle Sales Cloud.

In this chapter, you will learn about:

- Each sales and opportunity management object, and how to extend those objects using Oracle Application Composer
- How to modify sales and opportunity management objects using Page Composer

Extending Pages for Opportunities: Explained

Using Application Composer, you can change many items that appear in the pages for opportunities.

Use Application Composer to configure these pages:

- Opportunities landing page
- Create Opportunity page
- Edit Opportunity page
- Products table in the Edit Opportunity page
- These subtabs within the Edit Opportunity page:
  - Activities
  - Contacts
  - Deal Registrations
  - Leads
  - Notes
  - Partners
  - Team

For example, you can:

- Hide or show standard fields.
- Hide or show standard actions and buttons.
- Add user-defined fields of all types.
- Set fields as required.
- Set field default values.
• Reorder columns.
• Change column labels.
• Add user-defined actions and buttons.

High-Level Steps to Modify Opportunity Pages
The following are the high-level steps to make changes to the opportunity UI pages:

1. Activate a sandbox.
2. Navigate to the Application Composer, available under the Configuration menu.
3. Expand the **Opportunity** object tree.
4. In the Opportunity tree, click the **Pages** link.
5. Ensure that the **Simplified Pages tab** is active.
6. In the region that corresponds to the page you are modifying, duplicate the standard layout and edit the resulting layout. For example, in the Details Page Layout region, duplicate the standard layout and then edit the resulting layout. See the Configuring Opportunity Pages section in this topic for details about which regions in the Simplified Pages tab correspond to which opportunity pages.
7. In the new user-defined layout row, the Active check box will be automatically checked, indicating that the new user-defined layout is active.

**Note:** Since you have duplicated the standard layout, two layouts now display. When the application finds more than one layout, it evaluates each layout at run time, starting with the first layout in the table. If all conditions are met for the layout, then it is displayed. If not, then the next layout is evaluated, and so on.

8. Save your changes.
9. Publish the sandbox.

Configuring Opportunity Pages
The following table shows the opportunity pages or subtabs that can be modified, along with the navigation steps to the areas where you make the modifications.

All of these navigation steps assume the use of the Simplified Pages tab.

<table>
<thead>
<tr>
<th>Opportunity Page</th>
<th>Navigation</th>
</tr>
</thead>
</table>
| Landing (list) page | 1. Expand the **Opportunity** object.  
2. Click the **Pages** link.  
3. In the Landing Page Layouts region, duplicate the standard layout and edit the resulting layout. |
| Create Opportunity Page | 1. Expand the **Opportunity** object.  
2. Click the **Pages** link.  
3. In the Creation Page Layouts region, duplicate the standard layout and edit the resulting layout. |
| Edit Opportunity Page, Summary region | 1. Expand the **Opportunity** object.  
2. Click the **Pages** link.  
3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.  
4. Ensure the **Summary** subtab is active.  
5. Edit the Summary region. |
<table>
<thead>
<tr>
<th><strong>Opportunity Page</strong></th>
<th><strong>Navigation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edit Opportunity Page, Products table</strong></td>
<td>Optionally, first add user-defined fields to the Opportunity Revenue object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Edit the <strong>Edit Revenue Table</strong> region.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Contacts subtab</strong></td>
<td>Optionally, first add user-defined fields to the Opportunity Contact object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the <strong>Contacts</strong> subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Contacts subtab.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Team subtab, Team view</strong></td>
<td>Optionally, first add user-defined fields to the Opportunity Team Member object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the <strong>Team</strong> subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Fuse Opportunity Team region.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Team subtab, Territories view</strong></td>
<td>Optionally, first add user-defined fields to the Sales Territory object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the <strong>Team</strong> subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Territories region.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Partners subtab</strong></td>
<td>Optionally, first add user-defined fields to the Opportunity Partner object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the <strong>Partners</strong> subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Partners subtab.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Deal Registrations subtab</strong></td>
<td>Optionally, first add user-defined fields to the Opportunity Deal object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the <strong>Opportunity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the <strong>Pages</strong> link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the <strong>Deal Registrations</strong> subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Deal Registrations subtab.</td>
</tr>
<tr>
<td><strong>Edit Opportunity Page, Activities subtab</strong></td>
<td>You must use the Activity object in Application Composer to extend this subtab.</td>
</tr>
<tr>
<td></td>
<td>1. Navigate to the <strong>Activity</strong> object.</td>
</tr>
<tr>
<td></td>
<td>2. Click <strong>Pages</strong> and ensure the <strong>Simplified Pages</strong> tab is active.</td>
</tr>
</tbody>
</table>
### Extending Pages for Opportunities Using Application Composer: Worked Example

This example demonstrates how you can modify pages for Opportunities using Application Composer.

#### Extending Pages for Opportunities Using Application Composer

1. Navigate to the Application Composer Overview page.
2. In the object tree, select a standard object that includes a set of pages, such as Opportunity.
3. Select the Pages node.
4. Select the Simplified Pages tab.
5. Use the links on the tab to navigate to the object’s configuration pages, where you can modify the pages that are available for the selected object. You can show or hide fields, rearrange fields, and add your own fields.
6. For example, if you want to add the field ‘Budget Amount’ to the Create Opportunity page, select a layout in the **Creation Page Layouts** region. Click the **Edit** icon or select **Actions > Edit Layout**
7. In the **Edit Simplified Page Creation** page, select the **Edit** icon in the **Opportunity Create** region
8. In the Configure Detail Form region of the Opportunity Create page, move the **Budget Amount** field from the Available Fields to the Selected Fields list.
9. Click Save and Close.

**Related Topics**
- Extending Simplified Pages: Overview

## Configuring Fields in Simplified UI Pages Using Page Composer: Worked Example

Use this example to learn how to configure fields in Oracle Sales Cloud simplified UI pages using Page Composer. In this example, you learn how to configure the **Primary Competitor** field on the Edit Opportunities page for users with the Sales Representative job role. Configurations covered include hiding the field, make it required, and changing its label.

Before you configure pages in Page Composer, you must sign in as a user with a Sales Administrator job role or as a user with administrator privileges, and you must work in an active sandbox. A sandbox prevents your configurations from affecting the application until you publish the sandbox.

### Creating and Activating a Sandbox

1. Click on the user name and select **Manage Sandboxes...** from the menu. The Manage Sandboxes window appears.
2. If no sandbox is listed or if you want to keep your configurations separate from other configurations, then create a new sandbox:
   a. Click **Create**.
   b. Enter a sandbox name and optional description.
   c. Click **Save and Close**.
3. In the Manage Sandboxes window, select a sandbox from the list and click **Set as Active**. The name of the active sandbox appears on the page to the left of the user name.

### Configuring the Field

1. To configure a field in the Edit Opportunity page, click **Opportunities** to display the list of opportunities.
2. Click the user name or image, and select **Customize Pages** from the Settings and Actions menu to open Page Composer.
   
   **Tip:** When you are configuring UI pages, always open Page Composer while you are on the list page, before you navigate to the create or edit pages you want to configure.

   The Customize Pages window appears.
3. Select the layer that you want to configure. In this example, you want to configure the page for salespersons, so select the **Edit** option for the Job Role layer and select **Sales Representative** from the **Value** list.
4. Click **OK**.
The Page Composer toolbar appears at the top of the page indicating that you are editing the page with the Add Content mode selected.

The following figure shows a partial screen capture of the Page Composer toolbar with the Add Content mode selected.

![Page Composer toolbar screenshot]

**Note:** You must remain in the Add Content mode until you navigate to the page you want to configure.

5. Edit an opportunity by selecting its name link.

   The Edit Opportunity page appears.

6. Click **Select** in the Page Composer border to enter the Select mode.

7. Move your cursor over the field you want to configure, in this example, **Primary Competitor**, until the border appears and click.
The application displays a menu with two options: Edit Component and Edit Parent Component. The following figure shows a partial screen capture of the page with the menu open.

8. Select **Edit Component**.

The Component Properties window appears. Not every field supports all configurations, so the content of this window varies. You can make the following changes. Numbers refer to the figure that follows.

- To make the field read-only, select the **Read Only** option (1).
- To make the field required, select the **Required** option (2).

**Note:** Before making a field required, the field must display a value. If it does not, then cancel out of this window and enter the value first.

- To hide the field, deselect the **Show Component** option (3).
- To change the field label:
  - Click **Label** (4).
  - Select **Select Text Resource** from the menu.
  - Enter a name in the **Key** field. You can use this name to search for this label when you configure other pages.
  - Enter the new field label in the **Display Value** field.
  - Click **OK**.
The following figure shows a screen capture of the Common Properties window.

9. After you complete your configurations in the Component Properties window, click Apply and click OK.

Your configurations are visible on the page while you are in Page Composer.

10. Click Close to close Page Composer.

Note: Your configurations are no longer visible on the page because you are not signed in as a salesperson.

11. You can review the changes on the page while still in the sandbox by signing out and signing in again as a salesperson. To make the configurations permanent, you must publish the sandbox.

Modifying Competitors Pages: Explained

You can use Application Composer to modify the Competitors pages.
Competitors pages are available from the Navigator by selecting the Competitors menu item.

Note: To make only minor UI changes to the Competitors pages, use Page Composer instead of Application Composer.
Sales Competitor Object Pages and Navigation

To modify the Sales Competitor object and pages, you need to know which pages and regions can be modified. The following table lists the extensible pages and regions for the Sales Competitor business object, along with the navigation to the configuration pages. The Competitors module does not have pages marked as Simplified UI in Application Composer. Therefore, you use the Desktop Pages tab in Application Composer to modify the UI.

<table>
<thead>
<tr>
<th>Competitors Page</th>
<th>Competitors Region</th>
<th>Navigation</th>
</tr>
</thead>
</table>
| Competitors Overview page        | Competitors list region | 1. Select the Sales Competitor object.  
2. Click the Pages link.  
3. Click the Desktop Pages tab.  
4. Click the Edit Summary Table link. |
| Create Competitor page           | Details region      | 1. Select the Sales Competitor object.  
2. Click the Pages link.  
3. Click the Desktop Pages tab.  
4. Click the Edit Creation Page link. |
| Edit Competitor page             | Details region      | 1. Select the Sales Competitor object.  
2. Click the Pages link.  
3. Click the Desktop Pages tab.  
4. Click the Edit Summary Form link. |

Note: You must work in an active sandbox to be able to access the modification pages.

Creating New Competitor Fields

If you want new fields for competitor pages, you must first create them. The following procedure shows how to add a text-type field. For more details about adding fields, see the topics on adding fields using Application Composer.

1. Activate a sandbox using your company’s process.
2. Sign in as a setup user.
4. In the Objects tree, under Standard Objects, expand the object tree for the object you are modifying. For example, expand the Sales Competitor tree.
5. Click the Fields node for the object.
6. Click the Custom tab.
7. Click the Create icon, or select Create from the Actions menu. The Select Field Type page appears.
8. Select the type of field you are creating. For example, select Text as the type for a text entry field.
9. Click OK.
10. In the Appearance area, enter a Display Label for the new field. Set other optional attributes as desired.
11. In the Name area, enter an internal name for the field. This value is prepopulated based on your entry in the Display Label field, but you can override the default value. Ensure there are no spaces in the internal name.
12. Optionally, in the Constraints area, enter additional parameters for the field. For example, make the field a required field.
13. Optionally, in the Default Value region, enter a default value for the field.
14. Click **Save and Close**.
15. Publish the sandbox following your company’s process.

### Modifying Competitor Pages

Use the following procedure to modify competitor pages or regions.

1. Activate a sandbox using your company’s process.
2. Sign in as the sales administrator or as a setup user.
3. Navigate to **Configuration > Application Composer**.
   
   The Application Composer overview page appears.
4. In the Objects tree, under Standard Objects, expand the **Sales Competitor** object tree.
5. Click the **Pages** node for the Sales Competitor object.
6. All competitor pages are located under the Desktop Pages tab. Click the **Desktop Pages** tab.
7. Make the required changes. For additional navigation information, refer to the section in this topic, Sales Competitor Object Pages and Navigation.
8. Save your changes.
9. Publish the sandbox following your company’s process.

### Duplicate Display Names for Fields

When you configure Competitor pages, in the Available Fields list, some field display names appear to be duplicates. For example, you may see two fields for Primary Competitor. The names you see in the list are display names only; each has an underlying name, which is its actual name. To see the underlying name of the field, hover your mouse pointer over the field name. The hover text displays the actual name of the field.

**Related Topics**

- Extending Pages for Households: Explained
- Configuring Opportunity Pages Using Application Composer: Explained

### Modifying Your Object to Validate Assessments: Worked Example

Use this example to learn how to modify your account, lead, contact or opportunity to validate that a specified assessment template is completed, using Groovy scripts in Application Composer.

In this example, you are a sales administrator and you want to ensure that sales representatives complete the assessment template Discount Eligibility before they set an opportunity to Won.

### Validating Assessment Using Scripting

To ensure that sales representatives always fill in the Discount Eligibility Assessment template before closing an opportunity:

1. Sign in as a sales administrator.
2. Create and activate a sandbox.
3. Navigate to Application Composer.
4. From the **Applications** list, select **Sales**.
5. Expand **Standard Objects**, and then expand **Opportunity**.
6. Click **Server Scripts**.
7. On the Server Scripts Opportunity page, select **Triggers**.
8. In the Object Triggers region, click the **Add a new Trigger** icon.
9. On the Create Object Trigger page, select **Before Update in Database** from the **Trigger** list.
10. Enter a trigger name.
11. Enter the following script in the Trigger Definition region.

```java
def optyStatusCode = getAttribute('StatusCode')
if (optyStatusCode == 'WON') {
    def id = getAttribute('OptyId')
    def vo = newView('AssessmentVO')
    def vc = vo.createViewCriteria()
    def vcr = vc.createRow()
    def vci = vcr.ensureCriteriaItem('AssessedObjectId')
    vci.setOperator('=')
    vci.setValue(id)
    vc.add(vcr)
    vo.appendViewCriteria(vc)
    vo.executeQuery()
    def completedMandatoryAsmnt = false;
    if (vo.getEstimatedRowCount() > 0) {
        vo.reset()
        while (vo.hasNext()) {
            def row = vo.next()
            if (row != null) {
                def tempName = row.getAttribute('TemplateName')
                if (tempName == 'Discount Eligibility') {
                    def status = row.getAttribute('StatusCode')
                    println("Asmnt Status is " + status)
                    if (status != 'COMPLETED') {
                        throw new oracle.jbo.ValidationException('Please complete the Mandatory Discount Eligibility Assessment before changing status to WON')
                    } else {
                        completedMandatoryAsmnt = true;
                        break;
                    }
                }
            }
        }
    }
    if (completedMandatoryAsmnt == false) {
        throw new oracle.jbo.ValidationException('Please create and complete Mandatory Discount Eligibility Assessment before changing status to WON')
    }
}
```

12. Click **Save and Close**.
The trigger script displays a message whenever a sales representative changes the opportunity status to Won. This message ensures that the sales representative fills in the Discount Eligibility assessment template before changing the status to Won.
Hiding and Reordering Opportunity Fields Using Page Composer: Worked Example

You can hide fields, make fields required, and change the order of fields in the edit opportunity page, for specific job roles. You use Page Composer to do the modifications.

Prerequisites

Before you begin, consider the following setup requirements or prerequisites:

- This procedure applies only to the desktop UI.
- You must have the job role of a setup user or Sales Administrator to do this task.
- You must know how to create and publish sandboxes following your organization’s established processes. As a best practice, your organization might want you to first make changes in a sandbox before rolling the changes out to all users with the Sales Representative job role by publishing the sandbox.

Starting Page Composer

First, start Page Composer’s design mode to be able to make the changes:

1. Sign in to the application as a user with the Sales Administrator job role. Or, sign in as a setup user.
2. From the Navigator, select Opportunities.
3. Find an opportunity and edit it.
4. In the global region, expand the Settings and Actions menu which is available next to your user name. Then select Customize Opportunities Pages, under the Administration subheading
   The Customize Opportunities Pages dialog appears.
5. In the Customize Opportunities Pages dialog box, select the Edit option for the Job Role layer.
6. In the Value column next to Job Role, select Sales Representative from the list.
7. Click OK.
   The page opens in Page Composer design mode.
8. Next, toggle Page Composer to Direct Selection mode: Click the Select icon in the global region.
   Direct Selection mode is now invoked, allowing you to make field-level changes.

Hiding a Field and Changing the Order of a Field

To hide the Worst Case field and change the order of the Attachments and Partners fields, use the following steps:

1. With Page Composer in Direct Selection mode, in the Additional Details region of the edit opportunity page, hover over the Worst Case field.
A colored box appears around the field and its label.

2. Click inside the box around the **Worst Case** field.

A dialog box opens with two options: *Edit Component* and *Edit Parent Component*.

3. Select the **Edit Parent Component** option.

The Component Properties: panelFormLayout window opens.

4. In the Component Properties: panelFormLayout window, clear the check box next to the **Worst Case** field.

The following figure shows the Component Properties dialog box with Worst Case selected.

5. Select the down arrow to the right of the **Attachments** field to move it below the **Partners** field.

The order of the two fields changes.

6. Click **OK**.
Making a Field Read-Only

To make the **Win Probability** field read-only, use the following steps:

1. While still in Page Composer Direct Selection mode, hover over and then click the **Win Probability (%)** label.
   
   A dialog box opens with two options: Edit Component and Edit Parent Component.

2. Select the **Edit Component** option.

   The Component Properties: Win Probability (%) window opens.

3. In the Component Properties: Win Probability (%) window, select the **Read only** check box.

   The field becomes read-only, as shown in the following figure.

4. Click **OK**.
Making a Field Required and Not Sortable

To make the Quantity column in the Revenue Items table required and not able to be sorted, use the following steps:

1. With Page Composer still in Direct Selection mode, in the Revenue Items region, hover over the **Quantity** column and click its header.

   A dialog box opens two options: **Edit Component** and **Edit Parent Component**.

2. Select the **Edit Component** option.

3. Select the **Show Required** check box to make it a required field.

4. Clear the **Sortable** check box to make the column appear as not sortable.

   The following figure shows the Change Property dialog box with the Show Required and Sortable check boxes.

5. Click **OK**.
Committing Your Changes

When you are ready to commit your modification changes to the main line, perform these steps:

1. Click the Close button in the global region to sign out of Page Composer editor.
2. In the global region, expand the Settings and Actions menu, which is available next to your user name. Then select Manage Sandboxes, under the Administration subheading.
3. Select the row of the sandbox where you built your modifications.
4. Click the Publish button to commit your changes.

Verifying Your Changes

Verify your modifications by using the following steps:

1. When you are done with your changes, click Close in the header to sign out of Page Composer.
2. Sign out of the application.
3. Sign as a user with the Sales Representative role.
4. Navigate to the opportunity record that you edited, and verify the following:
   - The Additional Details region is expanded by default.
   - The Worst Case field is not visible.
   - The Win Probability (%) field is read-only.
   - The Attachments field is below the Partners field.
   - The Quantity column is marked as required with an asterisk, and it is not sortable.
21 Extending Sales Forecasting

Overview

Read this chapter to learn how to extend forecasting objects and pages in Oracle Sales Cloud. In this chapter, you will learn about each forecasting object, and how to extend those objects using Oracle Application Composer.

Related Topics

• Configuring and Extending Oracle Sales Cloud Sales Forecasting Pages: Explained

Configuring and Extending Simplified Pages for Sales Forecasts: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for sales forecasts. Use Application Composer to modify these pages:

• Sales Forecasts landing page
• Forecast Details page
  ◦ Products subtab
  ◦ Forecast Items subtab

When you modify the Forecast Details page, you will see two versions, or layouts, of the page in Application Composer. Each layout is targeted at a different user:

• The Edit Forecast page is available to sales managers who own a forecasting territory that is a parent of child territories. You can modify both the Products and Forecast Items subtabs using this layout.
• The Review Forecast page is available to salespeople who own a forecasting territory that has no child territories. You can modify only the Forecast Items subtab using this layout.

The Analytics subtab and Unforecasted Pipeline subtab are also available from the Sales Forecasts functional area, but are not extensible.

Sales Forecasts Landing Page

The Sales Forecasts landing page has two views that you can modify in Application Composer:

• The card view
• The summary table view

You can modify the region that appears at the bottom of the card view. You can modify the following items in this region

• Hide or show existing fields.
• Add custom fields (all types).

This figure shows the Sales Forecasts landing page card view, with the configurable region highlighted at the bottom of the page.

Note that you can’t make changes to these components on the page:

• Embedded analytics
• Custom buttons and actions

This figure shows the landing page table view. You can configure the fields in the table.
Products Subtab
The Products subtab includes the Edit Forecast Product Items table. You can modify the following items on the Products subtab in the Edit Forecast page:

- Hide or show existing fields.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
- Add custom actions and links.

Forecast Items Subtab
The Forecast Items subtab includes the summary at the top of the page. The Edit Forecast Items Search Results table, and the Edit Forecast Items table. You can modify the following items on the Forecast Items subtab:

- Hide or show existing fields.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
- Add custom actions and links.

Note that you cannot make changes to these components on the page:

- Actions menu (cannot add or hide and show)
- Custom buttons and actions

Using Application Composer to Modify the Sales Forecasts Simplified Pages
Before modifying pages, create custom fields and make other changes using the Forecast Territory Details standard object. To modify the simplified pages for sales forecasts using Application Composer:

1. Navigate to the Application Composer Overview page.
2. In the object tree, select the Forecast Territory Details standard object.
3. Select the Pages node.
4. Select the Simplified Pages tab.
5. Use the links on the tab to navigate to the object’s configuration pages, where you can modify the simplified pages that are available.

<table>
<thead>
<tr>
<th>Simplified Page</th>
<th>Application Composer Configuration Page Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Forecasts landing page</td>
<td>Edit Forecast Card View</td>
</tr>
<tr>
<td></td>
<td>Edit Summary Table</td>
</tr>
<tr>
<td>Products subtab</td>
<td>Duplicate the standard layout in the Details Page Layouts section and edit it, or create a new details page layout. The Products subtab appears only in the Edit Forecast page. When working with the details page layout for the Edit Forecast page, navigate to the Subtabs Region and click the Products subtab to configure the Edit Forecast Product Items table.</td>
</tr>
<tr>
<td>Simplified Page</td>
<td>Application Composer Configuration Page Link</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Forecast Items subtab</td>
<td>Duplicate the standard layout in the Details Page Layouts section and edit it, or create a new details page layout. The Forecast Items subtab appears within both the Edit Forecast and Review Forecast pages. You can configure the subtab for both pages by editing the details page layouts for each page. When working with the details page layout for both the Edit Forecast and Review Forecast pages, navigate to the Subtabs Region and click the Forecast Items subtab to configure the Summary region, the Edit Forecast Items table, and the Edit Forecast Items Search Results table.</td>
</tr>
<tr>
<td>Custom subtabs</td>
<td>You can also add subtabs to display records from child or related objects, for example. Click the plus + icon to create subtabs.</td>
</tr>
</tbody>
</table>
Glossary

**action**
The kind of access, such as view or edit, named in a security policy.

**analysis**
A selection of data displayed in one or more views, such as a table or chart, to provide answers to business questions.

**analytics**
Business intelligence objects such as analyses and dashboards that provide meaningful data to help with decision making.

**briefing book**
A collection of static or updatable analyses or dashboard pages that you can download, print, and share with others.

**business object**
A resource in an enterprise database, such as an invoice or purchase order.

**chrome**
The set of visual elements (for example, header; expand and edit icons) around the perimeter of a component or task flow that enables users to act directly on the object.

**configuration**
A change to the predefined artifacts of the application. Configurations impact multiple users.

**context layer**
A level that represents the scope of users impacted by configurations. For example, all users or only those who meet specific criteria.

**dashboard**
A page that provides quick access to key tasks and summary information for various objects within a functional area of interest.

**dashboard**
A collection of analyses and other content, presented on one or more pages to help users achieve specific business goals. Each page is a separate tab within the dashboard.
**data security**
The control of access and action a user can take against which data.

**desktop page**
A page that’s optimized for extended periods of use with monitors.

**desktop user interface**
A user interface that’s optimized for extended periods of use with monitors.

**detailed report**
A comprehensive report that provides detailed information about the subject matter. When you link a detailed report to an infolet, users can click anywhere in the infolet area to drill down to that detailed report.

**extension**
A new artifact in addition to what’s predefined in the application, for example a new business object or page.

**feature choice**
A selection you make when configuring offerings that modifies a setup task list, or a setup page, or both.

**filmstrip**
The single strip of icons above a page that you can use to open other pages.

**flexfield**
A flexible data field that you can configure such that it contains one or more segments or stores additional information. Each segment has a value and a meaning.

**flexfield segment**
An extensible data field that represents an attribute and captures a value corresponding to a predefined, single extension column in the database. A segment appears globally or based on a context of other captured information.

**global header**
The uppermost region in the user interface that remains the same no matter which page you’re on.

**infolet**
A small, interactive widget on the home page that provides key information and actions for a specific area, for example social networking or your personal profile. Each infolet can have multiple views.

**job role**
A role, such as an accounts payable manager or application implementation consultant, that usually identifies and aggregates the duties or responsibilities that make up the job.
**mainline metadata**
The primary branch of metadata that a sandbox is published to. Once published, changes made in the sandbox become available to all users.

**Navigator**
The menu in the global header that you can use to open the work areas and dashboards that you have access to.

**object workflows**
Object workflows represent a series of automated tasks that are configured to run when a trigger condition is met.

**offering**
A comprehensive grouping of business functions, such as Sales or Product Management, that is delivered as a unit to support one or more business processes.

**panel tab**
A tab that provides supplemental information or functionality for the page. Each panel tab is on the right side of the page, has an icon as the tab label, and slides out when you open the tab.

**performance tile report**
A summary report that shows data in the small infolet format. When you add a performance tile report to an infolet, users can see summary information about the subject matter.

**personalization**
A change that users make to control the look or behavior of the application. Personalizations impact only the user making the change.

**privilege**
A grant of access to functions and data; a single, real world action on a single business object.

**report**
An output of select data in a predefined format that’s optimized for printing.

**role**
Controls access to application functions and data.

**run time**
The type of activities that users perform while they are in a running application.

**sandbox**
A testing environment that isolates untested code changes from the mainline environment so that these changes don’t affect the mainline metadata or other sandboxes.
scheduled process
A program that you run to process data and, in some cases, generate output as a report.

setup user
A user provisioned with the job roles and abstract roles required to perform implementation tasks.

simplified page
A page that’s optimized for performing quick and frequent tasks on any device.

simplified user interface
A user interface that’s optimized for performing quick and frequent tasks on any device.

site layer
Application changes made in this layer affect all users.

springboard
The grid of icons on the home page that you can use to open pages.

task flow infolet
An infolet that displays summary information about a task.

user layer
Application changes made in this layer affect only the user making the change.

work area
A set of pages containing the tasks, searches, and other content you need to accomplish a business goal.

WSDL
Abbreviation for Web Services Description Language. It is an XML format that provides a model for describing Web services.