# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>i</td>
</tr>
</tbody>
</table>

## 1 About This Guide
- Audience and Scope | 1
- Related Guides     | 2

## 2 Customizations and Extensions
- Overview           | 3
  - Run Time Customizations and Extensions | 3
  - Design Time Customizations and Extensions: Highlights | 7
- Personalization   | 8
- Customization Layers | 9
- Changing Field Display Labels: Explained | 13
- Business Process Models: Explained | 15
- Enabling the Testing of Customizations | 15

## 3 Customization Life Cycle
- Customization Life Cycle: Explained | 19
- Run Time Customizations | 20
- Sandbox Manager         | 23
- FAQs for Using Sandboxes | 34
- Moving Customizations   | 36
- Viewing and Deleting Customizations: Procedure | 45
- Advanced Customization Life Cycle Tasks: Highlights | 46
### 4 Adding Objects and Fields in Application Composer

- **Overview**
- Extending Oracle Sales Cloud: How It Works
- Customizing Sales Cloud Applications Across Application Boundaries: Explained
- Viewing Application Composer Customizations: Explained
- Working with Objects
- Managing Security for Custom Objects
- Working with Fields
- Actions and Links: Explained
- Direct Page Links: Explained
- Importing and Exporting Custom Objects: Explained
- FAQs for Using Application Composer

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
</tr>
<tr>
<td>47</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>111</td>
</tr>
<tr>
<td>117</td>
</tr>
<tr>
<td>125</td>
</tr>
<tr>
<td>126</td>
</tr>
</tbody>
</table>

### 5 Extending Simplified Pages

- **Overview**
- Customizing Oracle Sales Cloud Simplified Pages: Explained
- Customizing Simplified Pages Using Page Composer: Procedure
- Creating a Set of Simplified Pages for Custom Objects: Explained
- Configuring a Search and Select Dialog Box: Explained
- Working with Dynamic Page Layouts
- Working with Subtabs
- Enabling or Disabling Drill Down Fields: Explained
- Specifying Drill-Down Fields for Custom Dynamic Choice List Fields: Example
- Creating and Adding Custom Action Links to Simplified Pages: Worked Example
- Customizing Work Area Lists
- FAQs for Extending Simplified Pages

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
</tr>
<tr>
<td>131</td>
</tr>
<tr>
<td>133</td>
</tr>
<tr>
<td>135</td>
</tr>
<tr>
<td>137</td>
</tr>
<tr>
<td>138</td>
</tr>
<tr>
<td>157</td>
</tr>
<tr>
<td>172</td>
</tr>
<tr>
<td>175</td>
</tr>
<tr>
<td>176</td>
</tr>
<tr>
<td>181</td>
</tr>
<tr>
<td>184</td>
</tr>
</tbody>
</table>

### 6 Extending Desktop Pages

- **Defining Pages: Explained**
- Creating a Work Area: Explained
- Subtabs: Explained
- Subtab Types: Explained
- Tree Nodes: Explained

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>187</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>212</td>
</tr>
</tbody>
</table>
## Extending Mobile Pages

- **Overview**
- Oracle CX Enterprise Mobile
- Oracle Sales Cloud Mobile

## Using Groovy Scripts

- **Overview**
- Groovy Scripting: Explained
- Server Scripts: Explained
- Global Functions: Explained
- Accessing View Objects: Explained
- Supported Classes and Methods: Explained
- Using Groovy Scripting: Examples
- Calling REST Web Services from Groovy Scripts
- Calling SOAP Web Services from Groovy Scripts
- Runtime Messages: Explained
- Debugging Your Groovy Scripts: Explained
- FAQs for Using Groovy Scripts

## Creating Object Workflows

- **Overview**
- Object Workflows: Explained
- Object Workflows: Examples
- Object Workflows and Field Updates: How They Work Together
- Object Workflows and E-Mail Notifications
- Object Workflows and Tasks
- Object Workflows and Outbound Messages
- Object Workflows and Business Processes
- Configuring Object Workflows: Worked Example
- FAQs for Creating Object Workflows
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Creating Custom Subject Areas</td>
<td>377</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>377</td>
</tr>
<tr>
<td></td>
<td>Custom Subject Areas: Explained</td>
<td>377</td>
</tr>
<tr>
<td></td>
<td>Objects in Custom Subject Areas: Explained</td>
<td>378</td>
</tr>
<tr>
<td></td>
<td>Fields or Attributes in Custom Subject Areas: Explained</td>
<td>380</td>
</tr>
<tr>
<td></td>
<td>Measures in Custom Subject Areas: Explained</td>
<td>383</td>
</tr>
<tr>
<td></td>
<td>Date Leveling in Custom Subject Areas: Explained</td>
<td>384</td>
</tr>
<tr>
<td></td>
<td>Creating and Editing Custom Subject Areas: Explained</td>
<td>385</td>
</tr>
<tr>
<td></td>
<td>Configuring Custom Subject Areas for Segmentation: Points to Consider</td>
<td>391</td>
</tr>
<tr>
<td></td>
<td>Securing Custom Subject Areas: How It Works</td>
<td>393</td>
</tr>
<tr>
<td></td>
<td>Publishing Custom Subject Areas: Explained</td>
<td>394</td>
</tr>
<tr>
<td></td>
<td>Associating Custom and Standard Objects and Creating Reports: Worked Example</td>
<td>395</td>
</tr>
<tr>
<td></td>
<td>Extensibility and Reporting: Example</td>
<td>399</td>
</tr>
<tr>
<td></td>
<td>Custom Subject Areas: Frequently Asked Questions</td>
<td>403</td>
</tr>
<tr>
<td>11</td>
<td>Page Customization</td>
<td>405</td>
</tr>
<tr>
<td></td>
<td>Customizing Pages: Overview</td>
<td>405</td>
</tr>
<tr>
<td></td>
<td>Page Composer Overview</td>
<td>405</td>
</tr>
<tr>
<td></td>
<td>Page Content and Template Customization</td>
<td>414</td>
</tr>
<tr>
<td></td>
<td>User Interface Text Customization</td>
<td>427</td>
</tr>
<tr>
<td></td>
<td>Theme Customization</td>
<td>436</td>
</tr>
<tr>
<td></td>
<td>Infolet Customization</td>
<td>441</td>
</tr>
<tr>
<td></td>
<td>Design Time Page Customization</td>
<td>444</td>
</tr>
<tr>
<td>12</td>
<td>Using Page Composer in Oracle Sales Cloud</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td>Customizing Oracle Sales Cloud UIs: Points to Consider</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td>Can I use Page Composer to customize all elements on a page?</td>
<td>448</td>
</tr>
<tr>
<td></td>
<td>Customizing Oracle Sales Cloud Pages: Explained</td>
<td>450</td>
</tr>
<tr>
<td></td>
<td>Customizing Simplified Pages: Procedure</td>
<td>461</td>
</tr>
<tr>
<td></td>
<td>Personalizing Dashboards, Transactional Pages, and Landing Pages: Overview</td>
<td>463</td>
</tr>
<tr>
<td></td>
<td>Changing Page Layout Using Page Composer: Procedure</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>Making Columns Unavailable for Display in Desktop UI Search Results: Worked Example</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>Customizing Search Results Information: Worked Example</td>
<td>468</td>
</tr>
<tr>
<td></td>
<td>Customizing the Global Page Template: Explained</td>
<td>469</td>
</tr>
<tr>
<td></td>
<td>What do I do if Page Composer quites while I am navigating in the UI?</td>
<td>471</td>
</tr>
</tbody>
</table>
## 13 Understanding Analytics and Reports

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizing Sales Cloud Analytics and Reports: Overview</td>
<td>473</td>
</tr>
</tbody>
</table>

## 14 Navigator and Springboard Customization

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizing the Navigator and Springboard: Overview</td>
<td>475</td>
</tr>
<tr>
<td>Customizing the Navigator and Springboard</td>
<td>476</td>
</tr>
<tr>
<td>Creating Categories and Page Entries for Navigation: Procedure</td>
<td>476</td>
</tr>
<tr>
<td>Configuring Links for Page Entries: Procedure</td>
<td>478</td>
</tr>
<tr>
<td>Managing Categories and Page Entries for the Navigator and Springboard: Procedure</td>
<td>480</td>
</tr>
<tr>
<td>Defining Settings for Home and Navigation: Explained</td>
<td>481</td>
</tr>
<tr>
<td>FAQs for Navigator and Springboard Customization</td>
<td>482</td>
</tr>
</tbody>
</table>

## 15 Help Customization

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizing Help: Overview</td>
<td>483</td>
</tr>
<tr>
<td>Who can add and manage custom help?</td>
<td>483</td>
</tr>
<tr>
<td>Help File Customization</td>
<td>484</td>
</tr>
<tr>
<td>Customization of Getting Started Pages</td>
<td>496</td>
</tr>
<tr>
<td>Embedded Help Customization</td>
<td>497</td>
</tr>
</tbody>
</table>

## 16 Activities and Sales Orders Extensibility

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>499</td>
</tr>
<tr>
<td>Extending Simplified Pages for Activities: Explained</td>
<td>499</td>
</tr>
<tr>
<td>Extending Simplified Pages for Sales Orders: Explained</td>
<td>501</td>
</tr>
</tbody>
</table>
17 Account, Contact, and Household Extensibility

Overview ................................. 503
Extending Simplified Pages for Accounts: Explained .......................... 503
Extending Simplified Pages for Contacts: Explained ......................... 505
Extending Simplified Pages for Households: Explained ....................... 508
Extending Simplified Pages for Relationships: Explained ..................... 510
Extending Simplified Pages for Assets: Explained ............................... 510
Extending Simplified Pages for Assets: Worked Example ..................... 512
Customizing the Overview Subtab: Explained ..................................... 515
Enabling Multiple Addresses: Explained ............................................. 521
Adding Join Fields to the Sales Account Object: Worked Example ........ 523
Oracle Sales Cloud Common Objects for Desktop Page Configurability: Explained ................................. 527
Customizing the Customers Work Area in the Desktop UI Using Page Composer: Explained ......................... 532
Customizing Customer Information Display in the Desktop UI: Worked Example .............................................. 534
Changing the Label and Width of a Table Column Using Page Composer: Worked Example ......................... 540

18 Leads and Marketing Extensibility

Overview ................................. 547
Sales Lead and Sales Campaign ......................................................... 547
Marketing ......................................................................................... 561
Copy Map ......................................................................................... 569
### 19 Outlook Extensibility

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>577</td>
</tr>
<tr>
<td>Customizing Outlook Pages Using Application Composer: Explained</td>
<td>577</td>
</tr>
<tr>
<td>Customizing Outlook Pages Using Application Composer: Procedure</td>
<td>578</td>
</tr>
<tr>
<td>Editing List Views: Procedure</td>
<td>580</td>
</tr>
<tr>
<td>Customizing Outlook Forms: Explained</td>
<td>582</td>
</tr>
<tr>
<td>Editing Form Fields: Procedure</td>
<td>583</td>
</tr>
<tr>
<td>Creating a Form Section: Procedure</td>
<td>584</td>
</tr>
<tr>
<td>Creating a Child Section: Procedure</td>
<td>586</td>
</tr>
<tr>
<td>Creating Role-based Form Layout: Procedure</td>
<td>588</td>
</tr>
<tr>
<td>Synchronization Settings: Explained</td>
<td>589</td>
</tr>
<tr>
<td>Server Synchronization Filters: Explained</td>
<td>591</td>
</tr>
<tr>
<td>Setting up a Server-side Filter: Procedure</td>
<td>593</td>
</tr>
<tr>
<td>Creating Role-based Synchronization Filter: Procedure</td>
<td>594</td>
</tr>
<tr>
<td>Editing Scripts: Explained</td>
<td>596</td>
</tr>
<tr>
<td>Creating a Script: Procedure</td>
<td>597</td>
</tr>
<tr>
<td>Configuring Custom Objects for Outlook Pages: Explained</td>
<td>599</td>
</tr>
<tr>
<td>Adding Parent Custom Object for Outlook Pages: Procedure</td>
<td>600</td>
</tr>
<tr>
<td>Editing Parent Custom Object List View: Procedure</td>
<td>603</td>
</tr>
<tr>
<td>Editing Parent Custom Object Form: Procedure</td>
<td>604</td>
</tr>
<tr>
<td>Adding Custom Child Object: Procedure</td>
<td>606</td>
</tr>
<tr>
<td>Editing Child Custom Object Form: Procedure</td>
<td>609</td>
</tr>
<tr>
<td>Adding Child Object to the Parent Object: Procedure</td>
<td>609</td>
</tr>
<tr>
<td>Defining Many-To-Many Relationships for Child Objects: Procedure</td>
<td>612</td>
</tr>
<tr>
<td>Selecting Outlook Setup Options: Explained</td>
<td>617</td>
</tr>
<tr>
<td>Synchronizing and Validating Changes: Explained</td>
<td>621</td>
</tr>
</tbody>
</table>

### 20 Partner Relationship Management Extensibility

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>623</td>
</tr>
<tr>
<td>Extending Simplified Pages for Partners: Explained</td>
<td>623</td>
</tr>
<tr>
<td>Customizing Oracle Sales Cloud Partner Pages Using Application Composer: Explained</td>
<td>625</td>
</tr>
<tr>
<td>Customizing Partner Management Dashboard and Pages Using Page Composer: Explained</td>
<td>627</td>
</tr>
</tbody>
</table>

### 21 Territory Management Extensibility

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extending Simplified Pages for Territories: Explained</td>
<td>631</td>
</tr>
<tr>
<td>Customizing Oracle Sales Cloud Territory Management: Explained</td>
<td>632</td>
</tr>
</tbody>
</table>
# Quota Management Extensibility

Overview

Customizing Oracle Sales Cloud Quota Management: Explained

# Sales and Opportunity Management Extensibility

Overview

Extending Simplified Pages for Opportunities: Explained
Extending Simplified Pages for Opportunities: Worked Example
Customizing Fields in Simplified UI Pages Using Page Composer: Worked Example
Customizing Sales Pages: Explained
Customizing Opportunity Pages: Explained
Customizing Your Object to Validate Assessments: Worked Example
Customizing the Desktop UI Sales Dashboard and Pages Using Page Composer: Explained
Hiding and Reordering Opportunity Fields Using Page Composer: Worked Example
Customizing the Sales Dashboard in the Desktop UI: Procedure

# Sales Forecasting Extensibility

Overview

Extending Simplified Pages for Sales Forecasts: Explained
Customizing Oracle Sales Cloud Sales Forecasting Pages: Explained

# An End-to-End Configuration Example

Overview

Creating a Sandbox: Example
Creating an Object: Example
Adding Child or Related Objects: Example
Advanced Field and Object Properties: Example
Adding Subtabs: Example
Testing Custom Object Security Settings: Example
Preface

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

Oracle Applications Help

Use the help icon to access Oracle Applications Help in the application. If you don't see any help icons on your page, click the Show Help icon in the global header. Not all pages have help icons. You can also access Oracle Applications Help at https://fusionhelp.oracle.com.

Using Applications Help

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

Additional Resources

- **Community:** Use Oracle Applications 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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, see the Oracle Accessibility Program.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send 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 customization toolset. Administrators who customize 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 customization 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 customize 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 run time 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 easily edit the user interface composition at run time. In Oracle Sales Cloud, Page Composer is intended for simple user interface editing functions, such as showing and hiding regions, fields, and tables, changing the order of regions, or changing a dashboard page layout. You can also use Page Composer for adding or removing predefined content from the Resource Library. All changes are done and stored in the UI layer. Oracle Sales Cloud transactional pages and dashboards are enabled for run time customization using Page Composer. These pages and regions are delivered already enabled for page editing. Administration and setup pages are not Page Composer-enabled.

To learn more about Page Composer, refer to the Page Composer chapter in this guide. Also, review each product-specific chapter in this guide to learn how to use Page Composer to make application changes for individual product areas.

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 a custom subject area using a wizard available in Application Composer.
To learn more about BI Composer, refer to the Creating Custom Subject Areas 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 customizing Oracle Sales Cloud.

<table>
<thead>
<tr>
<th>Guide</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Sales Cloud Getting Started with Oracle Sales Cloud Customizations</td>
<td>Introduces you to user interface elements, user interface types, and simple, common customizations of Oracle Sales Cloud.</td>
</tr>
<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 Customizing the Applications for Functional Administrators</td>
<td>Describes the tools and concepts for customizing and extending the applications.</td>
</tr>
</tbody>
</table>

**Related Topics**

- Oracle Help Center
2 Customizations and Extensions

Overview

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

This chapter covers:

- How to personalize applications
- How to do runtime extensions and customizations
- How to do customizations using customization layers
- What you can customize and extend using which tool
- How to test role-specific customizations

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 a customization in one area, that customization 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 customize one application are the same tools and processes to customize all applications.

Run Time Customizations and Extensions

Run Time Customizations and Extensions: Examples

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

Customizing the UI

To customize the UI, use:

- The Customize User Interface Text page to edit text that appears on multiple pages. For example, you can change the term, supplier to vendor 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 customize simplified and desktop pages for other users. For example, you can:
  ◦ Add fields
  ◦ Add validation
  ◦ Change defaults
  ◦ 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.

Customizing Navigation
Use the Structure page to customize the Navigator and springboard. From the Navigator menu, select Structure under Tools.

Adding Custom Attributes to Business Components Using Flexfields
Most business components, except those in Oracle Sales Cloud products, support using flexfields to extend the object with custom attributes. Use a flexfield to create custom attributes without programming. The 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 the flexfield, and corresponds to a reserved column in the application database.

Customizing Reports and Analytics
Predefined analyses, dashboards, and reports help in meeting business intelligence requirements. But you can customize them (for example, change the layout) to fit specific business needs.
For more information, see the Creating Analytics and Reports guides relevant to your products.

Customizing Help
Use the Manage Custom Help page to:
  • Customize the help files in the application Help
  • Determine which help files to show in specific help windows

You can open the Manage Custom Help page from any help window, or from the help site.

Note: You must have the appropriate job roles to customize help.

Related Topics
• Help File Customization: Overview
• Flexfields: Overview
Tools for Customizations and Extensions: Critical Choices

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

- Page customization
- Branding customization
- Object customization
- Security customization
- Business intelligence customization
- Help customization

Note:
- The following tables present only the top customization tasks, not all tasks.
- The tables don’t list design time customizations and extensions that usually developers perform. These customizations and extensions aren’t available in Oracle Cloud implementations.

Page Customization

This table shows some types of customizations that you can make to pages, and the corresponding tools to use. You can customize only certain pages in Page Composer. If the customization that you want to make isn’t available in Page Composer, then developers can customize the pages using Oracle JDeveloper (not available in Oracle Cloud implementations).

<table>
<thead>
<tr>
<th>Customization</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>Customize a page title</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize a task list menu</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize dialog box content</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize attributes for a flexfield on a page</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize properties for user interface (UI) components on a standard page</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize the UI Shell template</td>
<td>Page Composer</td>
</tr>
</tbody>
</table>
Customization | Tool
--- | ---
Customize a text string wherever it appears across all pages | User Interface Text
Customize the look and feel of application pages | Appearance page
Change the announcements on the home page | Announcements page

### Branding Customization
This table shows some types of customizations that you can make to use your own branding logo, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Customization</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customize 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>Customize report layouts</td>
<td>Layout editor in the BI application or external applications such as Microsoft Word</td>
</tr>
</tbody>
</table>

### Object Customization
This table shows some types of customizations that you can make to objects, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Customization</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 Customization
This table shows a security customization that you can make to objects, and the corresponding tool to use.

<table>
<thead>
<tr>
<th>Customization</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 Customization
This table shows some types of customizations 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>Customization</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>
<tr>
<td>Customize report layouts</td>
<td>Layout editor in the BI application or external applications such as Microsoft Word</td>
</tr>
<tr>
<td>Create a report</td>
<td>The BI application</td>
</tr>
<tr>
<td>Customize analyses</td>
<td>Reports and Analytics pane or the BI application</td>
</tr>
<tr>
<td>Customize dashboards</td>
<td>The BI application</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Customization</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customize 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>Customize 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 customizations. For example, Application Composer and Business Process Composer.

Related Topics
- Flexfields: Overview
- Help File Customization: Overview
Design Time Customizations and Extensions: Highlights

Developers can customize and extend applications using design time tools, for example, Oracle JDeveloper (a comprehensive integrated development environment). Design time customizations and extensions include complex changes that developers need to deploy into the run time environment.

✦ Note: These customizations and extensions aren’t available in Oracle Cloud implementations.

Design Time Tasks

• To do design time customizations, use Oracle JDeveloper or other tools such as Oracle SOA Composer. Refer to the Oracle Fusion Applications Extensibility Guide for Developers.

  See: Design Time Customizations and Extensions

  ◦ You can determine what users can personalize.

  See: Configuring End-User Personalization

• You can also create a complete Java EE application and integrate it with existing applications.

  See: Oracle Fusion Applications Developer’s Guide

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. To personalize the UI, click your user name in the global area and use the Personalization menu items.

✦ 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
• Composer-based personalizations, where you can redesign aspects of a page. For example, you can use Page Composer to rearrange regions on a page and add or remove content
FAQs for Personalization

How can I retain minor personalizations after I sign out?
Whether minor personalizations (such as changing table layout and hiding regions) persist after you sign out, depends on the page you’re on. However, to retain all minor personalizations, make changes by clicking your user name in the global area and selecting **Edit Current Page** under **Personalization** (if available).

**Related Topics**
- Saving Searches for Searches with Multiple Criteria: Procedure

How can I restore a page to the default content and layout?
Click your user name in the global area and select **Reset to Default Content and Layout** (if available for the page you’re on) under **Personalization**.

What happens if I restore a page to the default content and layout?
You remove all changes that you had previously made to the page using **Edit Current Page** (under **Personalization**, which you open by clicking your user name in the global area). The reset affects objects at the page level, for example, dashboard regions that you added or moved around, and dashboard layout changes. Restoring doesn’t reset personalization changes made within the components of a page, for example, within a dashboard region.

Customization Layers

Customization Layers: Explained
Use the built-in customization layers in your application to make customizations that affect only certain instances or users of an application. Before you create customizations, select the layer in which you want to customize. Most of the customization tools provide a dialog box for selecting the layer for your customizations. If the dialog box to select a layer doesn’t appear before you customize, then by default your customizations are made to the site layer.

Available Layers
The customization layers available to an application depend on its application family. However, all applications have the following customization layers:
- Site layer: Customizations made in the site layer affect all users.
- User layer: All personalizations are made in the user layer. Users don’t have to explicitly select this layer as it’s automatically applied while personalizing the application.

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. With the default customization layers, the user layer is the tip layer. An object may be customized more than once, but in different layers. At run time, the tip-layer customizations take precedence. For example, say you customize in the site layer. You use Page Composer to add a region on a page. A user personalizes the same page to hide the region. In such a case, the user-layer customization takes precedence for that user at run time.
Storage of Customizations and Layer Information

Customizations 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 customization layer. The customization 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 customizations 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 customizations.

Working With Customization Layers: Examples

The following scenarios illustrate working with customization layers to ensure that the correct customizations or personalizations are available at run time to appropriate users. For example, the Sales application has a layer for job role.

When you customize an artifact, you can choose to make that customization available only to users with a specific role, for example, a sales representative.

Customizing Pages for Users with Specific Roles

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

Following are the prerequisites:

- Activate a sandbox.
- When you customize a page for a specific job role, that role must be assigned to you for you to test the customization 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 customization. In this case, select the role layer with the value, Sales Representative. While customizing, 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 customization 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 customization engine in the metadata service checks the repository for XML files matching the artifact and the given context. On matching, the customization 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 customization 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 customization 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 customization XML file is applied to the base document and is displayed only for a sales representative.

### Personalization

All users of the application can use the Personalization menu items to personalize certain pages. For example, you can:

- Move elements around on a page
- Hide elements
- Add available elements to a page

While you personalize a page, the customization 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 customization engine in the metadata services:

- Pulls the XML file with the sales representative customizations 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
- Role Provisioning and Deprovisioning: Explained
- Autoprovisioning: Explained
- How do I provision roles to users?

Selecting Customization Layers to Include: Examples

While customizing, when you use the dialog box to select the customization layer, you can also include lower layers to view customizations from those layers.

The following scenarios explain what happens based on your selected layers. For these examples, the available layers are Site, Territory, and Job Role.

What You See While Customizing
Suppose you choose to:
- Edit the Job Role layer and select Sales Representative as the value for that layer
- Include the Territory layer and select Southwest as the value

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

While customizing in Page Composer, you see customizations that apply to sales representatives in the Southwest territory, based on:

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

### What Your Customizations Apply To

No matter what you see while customizing, the customizations you make apply only to the edit layer, Job Role. For example, say a field is hidden in the Site layer but displayed in the Territory layer for Southwest. No customization exists for the field in the Job Role layer for sales representative. Since Territory is higher than Site, you see the field displayed while customizing in Page Composer. However, suppose you choose to hide the field as part of your customization. So, that customization applies to the Job Role layer for sales representatives. Users with other job roles in the Southwest territory may still see the field. However, Job Role is higher than Territory. So, no sales representatives in any territory 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 one of three tools: Customize User Interface Text Tool, Application Composer, or Page Composer. 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, use the Customize User Interface Text Tool. For changes on a single page, use Page Composer. For changes to all labels for a custom object, 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's 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, let's say 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'll 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.
Changing Display Labels Across Multiple Pages

The Customize 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 Customize 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.

Let's look at changing 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, log 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 Tools 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 Display Labels on a Page

Let's look at changing the display label Customer to Prospect, but only on a single page that sales reps use to manage their prospects. In this case, you'll use Page Composer.

In contrast to the Customize User Interface Text tool, Page Composer accesses only one label instance at a time and only through a rendered desktop or simplified UI page. Even though its use is limited to these UI channels, using Page Composer to change a label might result in changes to other UI channels, such as Mobile, because a display label instance can be the source of display label values on any UI channel.

Also, Page Composer might not have access to all the display labels on a given page. In contrast, the Customize User Interface Text tool has access to the greatest number of display labels on any page.

Changing Custom Object Display Labels

Let's look at changing all the display labels for a custom object, Trouble Ticket. In this case, you'll 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 Customize 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 Customize User Interface Text tool.

Business Process Models: Explained

The application is based on business process models that map out business flows. When you customize 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 Customizations

What's Required for Testing Customizations in the Sandbox

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

What's Required for Role-Specific Customizations

If you are creating customizations 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 customizations in the sandbox. For example, if you are creating a
custom 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 Custom Objects

If you are creating custom objects, then you must assign yourself the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. The application automatically generates this custom object role the first time you create a custom object in the application. Unless users have this role, they cannot view or test custom 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 custom objects, then you must also add the Custom Objects Administration role. You must select both the **Self-requestable** and the **Autoprovion** option for this role. This object role is required for all custom objects, so you want to provision it automatically for future to sales administrators.

   For details, see the Enabling Sales Administrators to Test Customizations 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 Customizations in the Sandbox

Modify the provisioning rules to make it possible for sales administrators to assign themselves the job roles they need for testing customizations in the sandbox. For viewing and testing custom objects, sales administrators must have the Custom Objects Administration (ORA_CRM_EXTN_ROLE) role. To test job role-specific customizations, 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 custom objects, then you must also add the Custom Objects Administration role. The application automatically generates this custom object role the first time you create a custom object. For this job role select all of the options: Requestable, Self-requestable, and Autoprovision. All users creating custom objects must have this role when creating custom objects.
   
   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 customizations in the sandbox. For example, an administrator testing UI customizations for sales managers, requests the Sales Manager job role or the equivalent custom role. If you are creating custom 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 custom 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 custom objects, you must add the Custom Objects Administration role.

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 Customization Life Cycle

Customization Life Cycle: Explained

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

While using tools such as Page Composer, you can make application customizations in a sandbox. Sandboxes store customizations in Extensible Markup Language files in a separate Oracle Metadata Services repository that’s available only when you work in that particular sandbox. You can make the changes in either of the following:

- A test-only sandbox (that is, the code in the sandbox is for testing only, and is never deployed)
- A sandbox that’s then published to the full test environment

Developers using design time tools, such as Oracle JDeveloper, can deploy their customizations directly to that environment, or they can publish to a sandbox. For more information on design time customization workflow, see the Oracle Fusion Applications Extensibility Guide for Developers.

Note: Design time customizations aren’t available in Oracle Cloud implementations.
Project managers can monitor, import, and export customizations. Then, users can test the entire environment with all customizations, as shown in the figure below.

**Tip:** For on-premise implementations, developers may allow users, having access to the Setup and Maintenance work area, to configure the customizations and extensions made to the application.

---

**Run Time Customizations**
Run Time Customization Workflow: Explained

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

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

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

Note that a flexfield sandbox is for testing only and can’t be published. Instead, you can deploy a flexfield to the full test environment using the flexfield UI. To test a flexfield configuration before deploying it to the full test environment, deploy it to a flexfield sandbox. The changes that you deploy to a sandbox are isolated from the full 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 full test environment.

You can also use the Manage Customizations dialog box to:

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

You can also upload others’ customization metadata files to your environment.

This figure illustrates the use of sandboxes while:

- Customizing pages, objects, and security using Page Composer and Application Composer
• Configuring flexfields

Viewing and Diagnosing Run Time Customizations: Points to Consider

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

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

1. Click your user name in the global area, 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 customizations for a user under 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 customizations using the Search field.

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

Sometimes, an administrator may need to view a personalization that was made by another end user. For example, suppose a user made an error while personalizing a page and that page is no longer displayed for the user. Because the user can’t open the page, the user can’t correct the error. In this case, the administrator can access the page, request to see the user’s changes, and delete those changes to restore the page to its original settings.

Page-Level Customizations

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

Related Topics
- Customization Layers: Explained

Sandbox Manager

Overview

When customizing 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 customizations in their own sandboxes
- Which types of customizations 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 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 customizations 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 customizations before their users ever see them.

Sandboxes in Oracle Public Cloud Services provide robust, ready to use functionality to help isolate and control customization efforts without impacting 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 customize 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 customizations in a sandbox and, once satisfied, publish them to the mainline code. When making changes in a sandbox, your customizations 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 customizations replace the mainline code application’s existing configuration.

Within this topic, the term customizing means to change an existing artifact, for example, adding a new field to an existing business object. Customizing also refers to changing what is displayed on a page, as well as creating a completely new artifact, such as a business object or page.

Note: Never make your customizations 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 (FSM).

Composers

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

- Application Composer: Customize 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: Customize pages.


Note: To make customizations to most tasks in Application Composer, you must be in an active sandbox.

Customizations Affect Metadata Services and the Database Layer

At a technical level, your customizations 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 customizations 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.

Sandbox Usage
Sandbox Usage
Sandboxes handle metadata customizations 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 customizations 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 customizations 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

**Using Sandboxes: Points to Consider**

In the customization run time 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.

You can create two types of sandboxes:

- Sandboxes that are intended for testing purposes only
- Sandboxes that are intended to be published

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.
Customizations 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 customizing 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 customize the same artifact or perform customization tasks that indirectly affect other shared files. For example:

- A direct conflict arises when different users attempt to customize 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 customization 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’ll 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 customizations 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/applcore/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 customizing 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 ADF Business Components customizations, sign out and sign in again after switching in or out of this sandbox. This process ensures avoiding any inconsistencies between the run time caches and the ADF Business Components definitions.
Managing Customizations Using Sandboxes: Explained

You can apply different types of customizations 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 customizations are stored in sandboxes and are validated before applying them to an application.

Types of Customizations in Sandboxes

Sandboxes can contain the following types of customizations:

- Metadata changes - These changes (such as non-flexfield user interface (UI) page customizations) 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 customize your application in run time, you must first create a sandbox and then use tools, such as Page Composer to make the customizations. 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 full test environment. After testing the application, you can move to the production environment. The customizations created in the sandbox will be migrated to the production environment and will be available to the users.

Don’t make customizations directly in the mainline metadata. Make all customizations first in the sandbox. When you make changes to an application at run time 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 only see the customization information of the existing customizations in the current mainline metadata. For example, suppose you do customization in a sandbox and publish it. Then, on creating another
sandbox for the next customization, you will see customization1 in the new sandbox because customization1 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 run time tools to customize the application. For example, you can customize objects and pages using Page Composer, which uses sandbox manager. Oracle Business Process Composer and Oracle SOA Composer are also run time customization tools, but they don’t use sandbox manager. They have their own mechanisms for handling customization changes.

Metadata sandboxes that you create using sandbox manager are available in Oracle JDeveloper while creating and deploying customizations for a deployed application in Oracle WebLogic Server. During deployment, the available sandboxes (except security sandboxes) appear in a selection list in Oracle JDeveloper. Oracle JDeveloper is not available in Oracle Cloud implementations. You can save, download, and import the metadata sandbox sessions as files into other applications.

Managing a Flexfield Sandbox

To create a flexfield-enabled sandbox, deploy one 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 run time 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 customizations 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

Sandbox: How They Work with Some Customizations and Features

This topic covers the considerations you must keep in mind when working on some of the features and customizations using Application Composer or 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 a customization 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. 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 code 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 (including 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

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 code application.

Using Sandboxes: Explained

Maintain sandboxes using the Sandbox Manager. 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.

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 (re-enter) 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 area.

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

Note that if you log out and login 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 area 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 should always log out from Oracle Public Cloud Services and login 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 customizations using Application Composer. To start making customizations 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 should not do this, because it can cause problems with your sandbox.

### Publishing a Sandbox

Completed customizations 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 customizations 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 your application customization work.

Deleting a Sandbox

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

Once you have tested your customizations, you then move those customizations 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 code.
- **Publish:** Publish a sandbox using extreme caution. Once a sandbox has been published, all existing sandboxes derived from the same mainline code 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 customizations. 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:

- **Activate a sandbox.**
- **Exit a sandbox.**
- **Publish a sandbox.**
Importing Sandboxes
Do not import sandboxes. This operation is reserved for Oracle internal development only.

Setting Up Sandboxes: Procedure
To make customizations to the application artifacts, you need to 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: Do not 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 name in the global area, 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.

Related Topics
- Hierarchy in Profile Levels: Explained

Publishing Sandboxes: Procedure
After completing the customizations in the sandbox, publish them to make them available in the application.

Prerequisites
Before publishing the customizations, test or validate the changes at run time 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 name in the global area, 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.

Customizations are stored as XML files in the Metadata Services (MDS) repository and are segregated by sandboxes. When you customize an application artifact, your changes typically impact 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 customizations on the same object are saved at different times, then the last saved customization replaces the other customizations. For example: Let’s assume user A and user B are working in a sandbox and both are updating the same object. Let’s call their customizations customization A and customization B, respectively. When both users save their customizations at the same time, only one of the customizations is saved, say customization A is saved. Customization B is not lost, but when user B saves customization B again, customization 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. Do not share your user names to perform customizations.

- 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 shouldn’t 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 need to 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 customizations 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 is not recommended.

Once 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, document those changes and then delete all private sandboxes.

How can I manage server exceptions while publishing a sandbox?
When publishing a sandbox, a server exception may be encountered. Follow the guidelines below based on the exception error encountered:

- 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 which are not published.
Before you delete a sandbox, you must first confirm that the sandbox is not active.
Caution: Deletion of partial content of a sandbox is risky; it is recommended that you do not use this option.

After you have tested your customizations, you must move those customizations 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.

Moving Customizations

Using Customization Migration to Move Customizations: Points to Consider

Use the Customization Migration page to create a set of all customizations and extensions made to an application environment. Then, download the customization set and upload it into another environment. The customization set includes customizations across all product families. To open the Customization Migration page, select Tools - Migration from the Navigator menu.

Contents of the Customization Set

The customization set includes:

- Customizations done using Application Composer, except the following customizations:
  - 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

- Customizations done using:
  - Page Composer
  - Appearance page
  - Structure page
  - User Interface Text page
  - Page Integration Wizard page

- 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 entities

- Changes in scheduled processes
- Customizations done using the Manage Oracle Social Network Objects task
• Changes in security settings made in Application Composer

> **Note:** Enterprise roles, new duty roles, and role hierarchy changes, which are made directly in Security Console aren’t migrated.

• Customizations 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 customizations done using the business intelligence tools only if the Disable BI for Customization Set Migration profile option is set to **No**.

The customization set doesn’t include personalizations.

While an upload or restore activity processes Presentation Services customizations, 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 doesn’t 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

Some important points to consider are:

• The customization migration doesn’t include code extensions, such as managed beans, that you implement in Oracle JDeveloper using the applications developer role. These code extensions are stored in the `app-inf/lib` and `web-inf/lib` directories and you must manually move the extensions.

> **Note:** Oracle JDeveloper isn’t available in Oracle Cloud implementations.

• The type of customizations across all applications that will be added to the customization set are selected by default on the Customization Migration page. You can’t change this selection.
• To prevent including in-progress customizations in the customization set, make your customizations in a sandbox. The customization set doesn’t include customizations from a sandbox until the sandbox is published.

You can use the Customization Migration page to move customizations and extensions from any source environment to any target environment. However, you must always perform your customizations and extensions in a full test environment. Then use the Customization Migration page to move these changes to a production environment. As customization set migration
doesn’t provide a merge capability, never customize or extend a production environment. When you import a customization 
set, the rows in the database that are not preconfigured are updated if a matching record exists. Otherwise a record is 
inserted.

The customization set doesn’t include all deletions. For example, the set does not include the removal of a customization 
document using the Manage Customizations dialog box. After you import a customization set into the target environment, 
you must examine the environment for any deletions that you must make manually. Similarly, the customization 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

### Creating and Applying Customizations Using a Customization Set: Procedure

Create a customization set to move customizations across all the product families of the application from one environment to 
another. You can export all customizations, such as those stored in Oracle Metadata Services repository, JEDI, CRM, and BI 
using the Outgoing tab of the Customization Migration page. You can then import them to the target environment using the 
Incoming tab. You can use a customization set to move customizations in a batch instead of moving them one by one.

**Prerequisites**

Before creating a customization 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 and Oracle JDeveloper (not available in Oracle Cloud implementations) customizations made in 
sandboxes are complete before they’re published. Before starting the export process, you must publish all complete 
customizations. All customizations 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 Disable BI for 
  Customization Set Migration profile option is set to No in source and target environments. To view this profile option, 
  open the Manage Profile Options task in the Setup and Maintenance work area.
- You have been granted the following privileges, which enable you to access the Customization Migration page:
  - `FND_MANAGE_OUTGOING_CUSTOMIZATION_SET_PRIV`
  - `FND_MANAGE_INCOMING_CUSTOMIZATION_SET_PRIV`
  
Contact your security administrator for details.
- You never make customizations in the target or production environment while applying customizations.

**Note:** If you make customizations to the production environment in emergency circumstances, you must make 
the same customizations to the test environment. Making the customizations to the test environment ensures 
that the customizations are included in the next customization migration.
- You don’t perform customizations in the source environment during the export process.
Creating Customizations

To create customizations:

1. In the source environment, select Tools - Migration from the Navigator menu.
2. From the Outgoing tab of the Customization Migration page, click Create Customization Set.

   **Tip:** If the Delete button appears for an existing customization set, click the button. This action removes the temporary files that are on the server from the previous customization set creation. You can’t create a customization set until the previous set has been deleted.

3. Provide a name for the customization set.
4. Optionally, type a description of the set.
5. Click Save and Close.
6. Periodically, click Refresh to view the current status of the set creation. Eventually, the status changes to Ready for Download.
   
   To see the detailed status of each customization type, expand Customization Details.

   The process runs asynchronously. You can close the dialog box and return to it later.
7. Click Download, and specify the name and location for the file, and click Save. Ensure that the downloaded file is a JAR file.
8. After you download the file on your local file system, click Delete to remove the temporary files that were created on the server.

Applying Customizations

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

1. Open the Customization Migration page in the target environment.
2. From the Incoming tab, click Browse, specify the name and location of the customization set file, and click Open.

   If the Browse button appears disabled, click Delete to remove the previously uploaded customization set from the environment and enable the Browse button.

3. When the status for the customization set is Ready to Apply Customizations, click Apply.
4. Periodically, click Refresh to view the current status of the Apply action.

   The process runs asynchronously. You can close the dialog box and return to it later.

   For Oracle Cloud implementations, if problems occur during an Apply action, log a service request using My Oracle Support at https://support.oracle.com.
5. Access the target environment and examine the environment for any deletions that you must manually make.
6. Deploy all flexfields that display a Patched status.
7. 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 customizations 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.

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

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

   a. Open the customization set in the Incoming tab of the Customization Migration page.
   
   b. Click Restore to revert to the state before the customization 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 customization 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 customization 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 customization 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 customization instance during the customization import process and applied to the target environment.

- You can create custom reports directly in the target environment. However, ensure that you create the custom reports and reference them to the existing custom subject areas. Don’t create the custom subject areas directly in the target environment.

- You can initiate customization export and import tasks only from the mainline metadata. If you initiate customization from a sandbox, the process doesn’t execute.

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

10. 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 Customizations: Points to Consider

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

- To move customizations 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.
- To import a customization into another environment. For example, a customization developer using Oracle JDeveloper may see customizations that other users do.

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

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Tools to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move all customizations to another application environment.</td>
<td>Customization Migration.</td>
</tr>
<tr>
<td>Move only Oracle Metadata Services customizations made to pages and the user interface to another application environment.</td>
<td>Enterprise Manager Cloud Control.</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>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>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>Move only lookups to application environment.</td>
<td>Setup and Maintenance work area. Move application standard lookups, application common lookups, or both.</td>
</tr>
<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 customizations to a file to help diagnose an issue.</td>
<td>Manage Customizations dialog box.</td>
</tr>
<tr>
<td>Export customizations to import them into an application workspace in Oracle JDeveloper.</td>
<td>Manage Customizations dialog box.</td>
</tr>
</tbody>
</table>
**Note:** Enterprise Manager Cloud Control isn’t available in Oracle Cloud implementations. Therefore, in Oracle Cloud implementations, to perform tasks that require use of Enterprise Manager Cloud Control, log a service request using your help desk at https://support.oracle.com.

### Downloading Customizations

Use the Manage Customizations dialog box to download customization 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 area, and select **Manage Customizations** from the Administration menu. You can use these files for diagnosing customization issues.

You can also download all customizations of a page for all layers (**AllCustomization.zip**) using the **Download Customizations 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 Customization Set Reports

After exporting customizations, you can view and download a customization set report that contains a list of all customizations available in a customization set. To do so, click **Content Read Me** from the Outgoing tab of the Customization Migration page. To open the Customization Migration page, select **Tools - Migration** from the Navigator menu.

The customizations in this report include all:

- Custom objects
- Custom fields
- Custom pages
- Business intelligence (BI) changes

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

**Related Topics**

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

### Migrating Global Search Configurations: Points to Consider

The customization set, which you create using the Customization Migration page, doesn’t include global search configurations. After importing a customization 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 Customizations: Explained

After you’ve finished creating customizations in Application Composer, you can export and import them across Oracle application instances on the same release. Use Customization Set Migration (CSM) to export and import customizations. Some of the supported customizations that you can export and import include object UI extensions, object server scripts, saved searches, workflows, global functions and so on. Do not create these supported customizations manually in the target application instance. Import these supported customizations from the source instance only.

**Note:** In the target application instance, you must create only unsupported customizations.

To view the customizations in the target application instance, you must first export your customizations from the source instance as compressed files in either .jar or .zip format. You can then import these compressed files in the target instance. The following sections provide an overview of exporting and importing customizations. For more information on using CSM, see "Using Customization Migration to Move Customizations: Points to Consider."

**Note:** In CSM, the changes are directly copied to the mainline metadata and no sandbox is involved.

Exporting Customizations

You can export customizations by creating a customization set using CSM as compressed files in either .jar or .zip format. When exporting, keep in mind the following points:

- Do not modify the compressed files.
- Do not move customizations from the target application instance back to the source application instance.

To export customizations:

1. In the target application instance, click **Navigator - Tools - Customization Migration**.
   The Customization Migration page opens.
2. Click the **Outgoing** tab.
3. Create a customization set and enter the details in the **Create Customization Set** window.
4. After the customization set is created, click **Download**.
5. Download the customization set to a local folder.
Importing Customizations

After you’ve successfully exported the customizations to a compressed file format, you can import the compressed file into the target application instance.

To import customizations:

1. In the target application instance, click Navigator, Tools, Customization Migration. The Customization Migration page opens.
2. Click the **Incoming** tab.
3. Enter the name of the customization set.
4. Click **Apply**.

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’ve 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 CSM for importing. For example, if you’ve 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 CSM.

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

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

Migrating FND Lookups

You can use CSM to migrate FND lookups.

Supported Application Composer Customizations and Best Practices: Explained

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

This topic describes the following:

- Supported customizations
- Unsupported customizations
- Best practices for using CSM

Supported Customizations

The supported customizations 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: Recreate manually in the target Oracle Sales Cloud application.
• Custom subject areas: Republish all custom subject areas in the target Oracle Sales Cloud application. See the “Publishing Custom Subject Areas: Explained” section in the Oracle Sales Cloud Extensibility Guide.
• FND lookups
• All reports, analyses, and dashboards
• Migrate using FSM tasks: Manage Standard Lookups, Manage Custom Lookups, and Manage Set-Enabled Lookups.

Unsupported Customizations
Customizations 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.

Customizations made outside Application Composer and not supported include:
• Sales Prediction Engine (SPE) 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 customizations.

You must create unsupported customizations manually in the target application.

Best Practices for Using CSM
Best practices for using CSM include:
• Do not manually create supported customizations in the target.
• Manually create customizations that are partially supported or not supported by CSM
• Use CSM 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 made in Authorization Policy Manager (APM) from the source to the target prior to using CSM.
• During an export or import, you should not make customization changes in the source or target instance.

Viewing and Deleting Customizations: Procedure
Use the Manage Customizations dialog box to view the customizations made to the application pages and to delete unwanted customizations. Click your user name in the global area, and select Manage Customizations from the Administration menu.
Deleting Customizations

To delete customizations:

1. On the page that contains the customizations, 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 customizations.
3. Click **Delete** for the customization document that you want to delete.

Advanced Customization Life Cycle Tasks: Highlights

You can perform design time tasks as part of the customization life cycle.

> **Note:** These customization tasks aren't available in Oracle Cloud implementations.

**Customization Tasks**

- The logging functionality for customization set migration is different from the standard logging functionality for your application. You can adjust the amount of detail to log customization migration without requiring a server restart. For more information, refer to the Oracle Fusion Applications Administrator’s Guide.

  See: Configuring ApplSession Log Levels

- Developers can create customization metadata using Oracle JDeveloper, and package and deploy customizations to the source application environment. For more information, refer to the Oracle Fusion Applications Extensibility Guide for Developers.

  See: Design Time Customization Workflow

  See: Using Oracle JDeveloper for Customizations
4 Adding Objects and Fields in Application Composer

Overview

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

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

- The concept of Web applications, and how you can complete some customization tasks across Web application boundaries in selected Sales Cloud applications
- 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 customizations. Refer to the table of contents for these other chapters.

Also refer to additional guides, available in the Oracle Sales Cloud documentation library, for specific examples and use cases about customizing your implementation of Oracle Sales Cloud.

- Getting Started with Oracle Sales Cloud Customizations
- Oracle Sales Cloud Groovy Scripting Reference

You can access Application Composer by selecting Application Composer from the Navigator menu, under the Tools category. Remember to always select the desired application from the Application choice list first, before making any changes. 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 customize and 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 run time 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 log 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 customizations (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.

**Customizations for Nondevelopers**

Application Composer hides the complexity of customization 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:

- Customizing objects by adding new fields, or create entirely new objects.
- Creating foreign key-based relationships between two objects.
- Customizing 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 Tools category. The first view of Application Composer is the main Overview page, which is the entry point into all your customization options.

Getting Started with Customization Tasks

From the main Overview page, select the application you want to customize using the Application choice list. Then:

- Use the object tree to select the object you want to customize.
Or, click the New icon to create a new object.

**Application Composer**

<table>
<thead>
<tr>
<th>Application</th>
<th>Sales</th>
</tr>
</thead>
</table>

**Objects**
- Custom Objects
  - Quote
  - Sales Tool
- Standard Objects
  - Forecast Item
  - Forecast Territory Details
  - Opportunity
  - Opportunity Contact
  - Opportunity Reference
  - Opportunity Revenue
  - Opportunity Team Member

- Use the links in main Overview page, also known as the local area, to select a customization task. Or, use the links in the Common Setup pane.

**Common Setup**
- Relationships
- Role Security
- Custom Subject Areas
- E-Mail Templates
- Import and Export
- Object Workflows
- Business Processes
- Global Functions

Change the selected application in the **Application** choice list at any time to customize another Sales Cloud application.

**Related Topics**
- Defining Pages: Explained
- Extending Simplified Pages for Households: Explained
Customizing Sales Cloud Applications Across Application Boundaries: Explained

When you customize applications using Application Composer, you always do so within the context of a Web application, either Sales or Common. Objects that are typically shared with other applications reside in the Common application; the remaining Sales Cloud objects reside in the Sales application. The existence of these two Web applications doesn’t generally matter; you can complete customization tasks that cross Web applications. If you’re in a Global Single Instance environment where multiple Cloud services are deployed, however, then there are some restrictions in terms of what you can customize across Web applications.

This topic introduces you to the concept of Web applications, and also describes the customization tasks that cross Web application boundaries.

**Note:** If you’re in a Global Single Instance environment where multiple Cloud services are deployed, read the section “Global Single Instance Environments” to understand the restrictions in place around crossing Web application boundaries.

Selecting a Web Application

To complete a customization task, such as create a new custom object, you first select an application (Sales or Common) on the main Overview page of Application Composer. The new custom object will belong only to the application that you select.

**Tip:** When you first open Application Composer, the default application is always Common. If you previously made customizations in another application, such as Sales, then you must manually change the application using the Application choice list to Sales before you can review and update those customizations. Web applications are also referred to as application containers.
Cross-Application Customization Tasks

A cross-application customization task refers to any task that you can do to an object in one Web application by accessing an object in another Web application. For example, you might want to add a subtab to an opportunity's page (Sales) that displays contact records (Common).

In general, customization tasks that are available in Application Composer can be categorized into three areas:

- **Object model customizations**
  
  For example, add a new field to an object, or create a new object entirely.

- **User interface customizations**
  
  For example, show or hide a field. Or create a work area for a new top-level object.

  **Tip:** A top-level object is an object that does not have a parent as part of its definition. In other words, it is not a child object. Top-level objects have their own work areas (a set of user interface pages such as an overview page and details page). A child object does not have its own work area, and appears instead as a subtab or tree node within the work area of its parent.

- **Scripting customizations**
  
  For example, make a field conditionally required based upon a value in another field.

You can make object model, user interface, or scripting changes that cross Web application boundaries.

**Note:** The Common application is available for cross-application customization tasks, provided that only Oracle Sales Cloud is implemented in your environment. See "Global Single Instance Environments" below.
Object Model Customizations

When customizing an object’s model, you can reference standard and custom objects from other Sales Cloud applications. The types of changes you can make include:

- Dynamic choice lists
  
  A dynamic choice list is a field that contains a list of values which are populated from the actual data of another object.
  
  From a cross-application perspective, you can create a dynamic choice list field for an object in one Web application that is populated by the records from an object in another Web application.

  - For example, create a dynamic choice list field for the opportunity object (Sales) which is populated with contact records (Common).

- Relationships
  
  A relationship is a foreign key association between two objects, and indicates a connection between two objects’ data. (You create a relationship between two objects, so that you can later expose this connection on user interface pages through the use of child or related object subtabs or tree nodes.)
  
  From a cross-application perspective, you can create a relationship between two objects from different Web applications.

  - For example, create a relationship where the opportunity object is the source and the contact object is the target.

  This enables the creation of the contact object subtab on the opportunity object’s details page. See the "User Interface Customizations" section in this topic.

  **Note:** Relationships enable not only user interface changes, such as subtabs and tree nodes. In addition:

  - Each object in the relationship becomes available for scripting against the other object. See the "Scripting Customizations" section in this topic.
  - Objects in a relationship are available to be selected as child objects, when defining a custom subject area.

User Interface Customizations

When customizing an object’s user interface, you can reference standard and custom objects from other Sales Cloud applications. The types of changes you can make include:

Subtabs and tree nodes: You can display details that are related to the current object but derived from another object. You do this by adding subtabs to an object’s details page, and specifying the source of subtab data, or by adding tree nodes to the object tree, and specifying the source of tree node data. From a cross-application perspective, the source of subtab or tree node data can come from an object in a different Web application.

- For example, add a Contacts subtab to the Opportunity details page.
- Or, add a Competitor tree node to the Customer 360 tree.

Note that a relationship must first exist between the two objects, prior to adding subtabs or tree nodes.
Scripting Customizations

When writing Groovy scripts as part of customizations to an object, you can reference standard and custom objects from across other Sales Cloud applications in your scripts. The types of changes you can make include:

When writing a script, use the newView() built-in function to access data from any standard object or custom top-level object in any Sales Cloud Web application. For more information on the newView() function, see "Accessing the View Object for Programmatic Access to Business Objects" in Oracle Sales Cloud Groovy Scripting Reference at http://docs.oracle.com/cloud/latest/saldescs_gs/CGSA/index.html.

Note that a relationship must first exist before you can script against other objects.

- For example, write a script in a "before update" trigger on the opportunity object that counts the number of contacts assigned to an opportunity. At run time, when a user saves an opportunity record, the trigger populates a custom field on the opportunity with the number of contacts.
- Or, write a script in a "before insert" trigger on the account object that retrieves information about the opportunity associated with the organization that the account is tied to.

Global Single Instance Environments

If you’re in a Global Single Instance environment where multiple Cloud services are deployed, then the concept of Web application "boundaries" is important because it directly impacts what you can do when making changes using Application Composer.

You can make object model, user interface, or scripting changes that cross Web application boundaries, provided that the object you are customizing exists in the Sales Web application. Here’s what you can do:

- You can reference all objects (both custom and standard) that exist in the Sales Web application.
- You can also reference standard objects (but not custom) from the Common application.

  Objects available in the Common Web application include common components such as tasks, interactions, and notes, as well as Master Data Management (MDM) and Common Party User Interface (CPU1) objects. These Common objects are available to you when customizing an object in the Sales Web application.

When customizing Common objects themselves, however, the object model and user interface cross-application customization tasks described above are not available. Scripting customizations are available, although you should proceed carefully when writing scripts for Common objects that access objects in the Sales Web application. Your scripts will work only when triggered within the context of Sales.

 américita: Object model customizations include the creation of custom objects, as well. When creating a new object, consider whether you need to share that object with any objects that exist under the Common Web application. If yes, then create that object as a top-level custom object in the Common Web application. If not, then create the object in the Sales Web application.

Related Topics

- Subtabs: Explained
- Tree Nodes: Explained
- Groovy Scripting: Explained
Viewing Application Composer Customizations: Explained

Use the Customization report to view a detailed list of Application Composer customizations 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.

Customization Report

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

The report includes:

- Customizations to standard objects
- Custom objects
- Custom fields added for both standard and custom objects
- Dynamic page layouts
- Global functions
- Object validations
- Object functions
- Object triggers
- Object workflows

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. Download the report by selecting the format, either HTML or Excel.

Working with Objects

Defining Objects: Explained

Using Application Composer, customize 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 customization. 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
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 customization 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 Tools category.
2. On the main Overview page, select an application from the **Application** choice list.
3. 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.

**Application Composer**

<table>
<thead>
<tr>
<th>Application</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects</strong></td>
<td></td>
</tr>
<tr>
<td>Custom Objects</td>
<td>![Custom Objects Icon]</td>
</tr>
<tr>
<td>Quote</td>
<td>![Quote Icon]</td>
</tr>
<tr>
<td>Sales Tool</td>
<td>![Sales Tool Icon]</td>
</tr>
<tr>
<td>Standard Objects</td>
<td></td>
</tr>
<tr>
<td>Forecast Item</td>
<td>![Forecast Item Icon]</td>
</tr>
<tr>
<td>Forecast Territory Details</td>
<td>![Forecast Territory Details Icon]</td>
</tr>
<tr>
<td>Opportunity</td>
<td>![Opportunity Icon]</td>
</tr>
<tr>
<td>Opportunity Contact</td>
<td>![Opportunity Contact Icon]</td>
</tr>
<tr>
<td>Opportunity Reference</td>
<td>![Opportunity Reference Icon]</td>
</tr>
<tr>
<td>Opportunity Revenue</td>
<td>![Opportunity Revenue Icon]</td>
</tr>
<tr>
<td>Opportunity Team Member</td>
<td>![Opportunity Team Member Icon]</td>
</tr>
</tbody>
</table>

**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 need 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. Select an application on the main Overview page of Application Composer. The new custom object will belong to the application that you select.

2. Select the **Custom Objects** node in the object tree, or click the icon in the local area of the main Overview page. On the resulting summary table, click the New icon.

3. Or, at the top of the object tree, click the New icon.

4. Complete the primary identifying attributes for a custom object:

   a. **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.

   b. **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.

   c. **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, let’s say 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.

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

      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:** You can use a custom object’s internal name only once across the mainline code and all existing sandboxes. If you’ve 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**

5. Click OK.

Once your custom object is created, you’ll want to 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. Select an application on the main Overview page.
2. To view the Overview page for an object:
   a. Select the object in the object tree.
   b. 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, 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 below.

• 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 above.

2. On the Object Overview page, click **Edit**:

   o Change the object’s primary attributes, such as display label, description, and record name, at any time.

   o 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’ll 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.
### Relationship Type

<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_ _Id_c.</td>
<td>&lt;Name of the parent object&gt;_ld_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_ _Id_c.</td>
<td>&lt;Name of the dynamic choice list field&gt;_ld_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, and the relationship name is relation_ <em>Mto1, then the foreign key API name (added to the target object) is SourceObj</em> <em>Id</em>&lt;Name of the relationship&gt;_Mto1.</td>
<td>&lt;Name of the source object&gt;<em>ld</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.

See: "Configuring a Search and Select Dialog for Custom Objects" below.

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_c and SourceObj_Id_Src_SourceObj_cToIntersectionObject_c.</td>
<td><code>&lt;Name of the object&gt;_Id_Tgt_&lt;Name of the object&gt;_cTo&lt;Intersection object name&gt;, &lt;Name of the object&gt;_Id_Src_&lt;Name of the object&gt;_cTo&lt;Intersection object name&gt;</code></td>
</tr>
</tbody>
</table>

**Related Topics**

- Configuring a Search and Select Dialog Box: Explained

**Activating Global Search for Custom Objects and Deactivated Application Objects: 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 custom objects you created. You can activate search only on custom objects, not on custom 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. Search for the Manage Search View Objects task.
3. Click the task name link in the search results.
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.
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 will want to add any custom fields you want to enable for searching, for example.

- The **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.
In the following 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.

- The 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 any custom fields 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
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 customizations) and the Custom Objects Administration job role and Sales Administrator job role (for testing the customizations 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’ve assigned the Custom Objects Administration role to yourself and to other users who create customizations.

   The Custom Objects Administration role is automatically assigned to new custom objects, as well as existing custom objects, if you’ve 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 customization 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’ll 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’ll 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’ll 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, you’ll want to 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 customizations to predefined roles in an earlier release. During the upgrade to the current release, Oracle automatically copies those customized 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 Customizations in the Sandbox
- Assigning Yourself Additional Job Roles Required for Testing
- Copying Sales Roles: Points to Consider
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 then select an application on the main Overview page.
3. Select a custom object in the object tree.
4. 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 select an application on the main Overview page.
3. 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.
4. 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 Tools category in the Navigator.
2. Select an application from the Application choice list on the main Overview page.
3. Expand the object that you want to add custom fields to.
4. 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

When you create a custom field in Application Composer, you select from a set of standard field types, such as a number or formula field. Each field type has a set of standard properties. For example, for all field types, you must specify a display label for the field. You can also indicate if the field that you’re creating is required. Most properties are common across all types of custom fields, although some are specific to the type of field you’re creating. Read this topic to learn about these common properties.

Before you create a new field for an object, you should understand:

- The set of standard 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
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 run time application using the Groovy-based expression included in the formula field’s definition. This is a read-only field that users in the run time 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 system automatically adds the percent sign.

- **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 Field Properties

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.

<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>Help Text</td>
<td>Appearance</td>
<td>Set this property for all field types.</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>Name</td>
<td>Name</td>
<td>Set this property for all field types.</td>
</tr>
</tbody>
</table>

Field Property:

- **Label**: Specify the display label for the field. You should limit the label to a maximum length of 80 characters, although no maximum length is enforced.

Field Property Region:

- **Appearance**: These properties control how the field appears to your users in the run time application.

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

Field Types:

- **Set this property for all field types**.
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.

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
<th>Related Field Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field names can contain only underscores and alphanumeric characters. They must:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Begin with a letter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Not contain spaces</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Not end with an underscore</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Not contain consecutive underscores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Not include special characters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This might cause issues while generating clients using SOAP Web services.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Be limited to a maximum of 28 characters if the characters are single byte.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If the characters are multibyte, such as Japanese or Chinese, then the maximum character limit is 28/number of bytes per multibyte character.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example, if characters are 2 bytes, then the name is limited to a maximum of 14 characters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If a mix of characters is used, then 28 is the maximum sum of character bytes that is supported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You cannot change this property after the field is created.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tip:</strong> 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The API name, used in your Groovy scripts, is also automatically generated for a field by taking the logical name and appending <code>_c</code>. 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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Set this property for all field types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A unique field description for internal use only.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Required</th>
<th>Constraints</th>
<th>Set this property for all field types except for formula fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicate if the field is required. You can also optionally use the expression builder to write</td>
<td>Specify constraints, which let you control the run time behavior of the field.</td>
<td></td>
</tr>
</tbody>
</table>
Field Property | Field Property Region | Related Field Types
--- | --- | ---
Field Property Region
- an expression that specifies the conditions that must apply for this field to be 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 run time.

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 run time.

The object’s Web services also reflect the required fields when your sandbox is published to the mainline metadata.

**Updatable**

Constraints
- Set this property for all field types except for formula fields.

Indicate if the field is updatable. You can also optionally use the expression builder to write an expression that specifies the conditions required for this field to be updatable. This includes being updatable on a desktop page, via Web services, through the import and export functionality, and by server scripts.

If you write an expression to control whether a field is updatable, 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 run time.

**Searchable**

Constraints
- Set this property for all field types except for long text and formula fields.

Indicate if you want this field to be made available for selection as an additional search criterion from the Add Fields choice list in the Advanced Search mode.

**Indexed**

Constraints
- Set this property for the following field types:
  - Text
  - Number
  - Date
  - Datetime
  - Currency

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

#### Field Property

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.

#### Constraints

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

- **Include in Service Payload**

#### Fixed Value

Specify a literal default value for the field.

- **Fixed Value**

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

#### Expression

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

- **Expression**

<table>
<thead>
<tr>
<th>Related Field Types</th>
<th>Percentage</th>
</tr>
</thead>
</table>

You cannot index the following field types:

- Long Text
- Formula
- Check Box
- Fixed Choice List
- Dynamic Choice List

### Related Field Types

- Percentage

Set this property for all field types except:

- Check Box
- Formula
- Fixed Choice List
- Dynamic Choice List

For these field types, write server scripts.
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 customizing and 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.

Related Topics
- Groovy Scripting: Explained
- Changing Field Display Labels: Explained

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>
<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>
Field Property | Field Property Region
--- | ---
Fixed Value | Default Value

**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 runtime 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><strong>Label</strong></td>
<td>Appearance</td>
</tr>
<tr>
<td><strong>Help Text</strong></td>
<td>Appearance</td>
</tr>
<tr>
<td><strong>Display Width</strong></td>
<td>Appearance</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Name</td>
</tr>
<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>Indexed</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 certain field types, including currency fields:

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expression</strong></td>
<td>Default Value</td>
</tr>
</tbody>
</table>

### Maximum Length

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. See below for setup instructions.

### 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 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>
</tbody>
</table>
Oracle Sales Cloud
Customizing 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>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:
Field Property | Field Property Region
--- | ---
Description | Name
Required | Constraints
Updatable | Constraints
Searchable | Constraints
Indexed | Constraints
Fixed Value | Default Value

**Note:** When you select a literal date for this field’s default value, the selected date appears in the international standard notation for date and time of day; however, at run time, the date and time of day are displayed according to the user’s preferences.

Expression | Default Value

### 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: 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 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>
<tr>
<td>Additional List Display Values</td>
<td>Additional List Display Values</td>
</tr>
<tr>
<td>Additional List Search Fields</td>
<td>Additional List Search Fields</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

  ![List Data Source](image)

  **List Data Source**
  Select the related object you want to query to populate the list. Then, select the related object field value you want to display for the user's selection.

  - **Related Object**
    - Account
  - **List Selection Display Value**
    - Name

  ![Data Filter](image)

  **Data Filter**
  Add search field criteria to filter the query of the related object specified above.

  - **Add Search Field**

  No search fields added.

- 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 run time. 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 run time by using data filters. Ideally, set data filters on a dynamic choice list during the initial configuration of the field.

In our example, we could filter out any accounts outside a particular region.

You can also filter out records that are inactive. In our example, we could filter out any accounts that are set to inactive. At run time, 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 run time by including one or more filters based on indexed fields.

• Additional List Display Values region

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** shuttle to include additional related object fields in the dynamic choice list at run time. These additional fields assist your users in making a selection from the choice list. The shuttle 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 shuttle to include additional related object fields in the dynamic choice list’s Search and Select dialog, accessed using the Search... link at run time. The shuttle 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 run time, 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>
<th>Pattern Used</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>
<td>&lt;Name of the dynamic choice list field&gt;_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.

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>
</tbody>
</table>

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

**Description:**

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><strong>Display Type</strong></td>
<td>Appearance</td>
</tr>
</tbody>
</table>

**Description:**

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.

**Note:** If you create a multiple-select fixed choice list, then do not use commas in the lookup codes that populate this field.
Field Property | Field Property Region
--- | ---
Lookup Type | List of Values

You cannot create a Lookup Type with a name ending in "LOOKUPTYPE". If you do, you won’t be able to see this extension in BI Answers and reporting.

Constrain List by Parent Field Value Selection | List of Values

## 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 (via product code), as well as all custom lookup types that have been created for your Sales Cloud implementation.

### List of Values

Configure the list of values you want to display in the choice list. Click the search icon to select a predefined lookup type, or create a new one.

- **Lookup Type**
- **TT Area**
- Constrain list by parent field value selection

Select the parent field whose value selection will drive the contents of this field, and then define the mapping between the parent and child values.

**Parent Choice List**

- **Type**

**Parent Lookup Type**

- **TT Type**

### 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’ll 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.
Field Property | Field Property Region
--- | ---
**Display Type** | Appearance

If the formula type is Text, then indicate if you want this formula field to render in the run time application as a simple text box, or if the field should allow multiple lines where text can wrap.

<table>
<thead>
<tr>
<th>Depends On</th>
<th>Constraints</th>
</tr>
</thead>
</table>

**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 an object and click the Joins node. For example, select the Sales application, 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 customization 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, available in the Common application.

• 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.

Related Topics
• Adding Join Fields to the Sales Account Object: Worked Example

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:

<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>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 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 point.
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 numerical value that a user can enter into this field.

**Maximum Value**

The maximum numerical 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>
</tbody>
</table>

The following properties are unique to only certain field types, including percentage fields:
### Field Property Region

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

**Decimal Places**

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**

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 | Field Property Region
--- | ---
Description | Name
Required | Constraints
Updatable | Constraints
Searchable | Constraints
Fixed Value | Default Value

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

Field Property | Field Property Region
--- | ---
Lookup Type | List of Values

Selecting the lookup type is possible only during field creation.

Available Record Types | Roles
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 | Roles
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 (via 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>
</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>

### Display Type
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.

### Maximum Length
The maximum number of characters that a user can enter in the field. You can set a maximum length of 1500 characters. If the field is a multiline field, then carriage returns are permitted and count as part of the total.

**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.

### Minimum Length
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).
- 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.

Actions and Links: Explained

In Oracle Sales Cloud, you can add actions, such as scripts, and buttons 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

An action can be based on a script (a Groovy method that is defined on the object) or a URL. After you create an action, it can be exposed as a button or an option on the Actions menu. After you create a link, it can be selected as a field for display at run time.

A button can perform an action or navigate the user to another page in the run time application, or to another Web site. For example, you might want to provide a static link from an overview page to a corporate Web site. Or, you might want to include a button on a summary table, which users can click at run time 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.

**Note:** 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.

When configuring the work area for a standard or custom object, you can add custom actions or links to a page-level or task-level Actions menu or as a toolbar button. 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 the Create and Edit subtabs and tree nodes.

**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 run time (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 run time, 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 run time the buttons are stacked and made available using a drop-down icon.
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 an Overview page or Details page.

The following figure shows a button and a link added to the Sales Opportunities Overview page.

1. Defining Actions or Links

To define an action or link for an object:

1. Select an application on the main Overview page.
2. Select a standard or custom object in the object tree.
3. Select the Actions and Links node.

To create a new script or URL:

1. In the Create Action or Link page, enter a descriptive name in the Display Label field.
2. For Type, select Action and, for Source, select Script or URL.
3. In the Script region click the New icon.
The following figure shows the Create Action or Link page showing a static URL enclosed in double quotation marks.

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.

The following figure shows a script in the URL Definition window.
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.

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.

> Note: 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.
The following figure shows a selected link and fields in the Edit Summary Table page in the Pages node for the Opportunity object.

When choosing to display a link, you select it just as you select to display standard or custom fields. This is because, at run time, the UI displays the URL link as if it is a field in a table. 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. Depending on how you configure actions and links, in the run time summary table, you could see both the buttons and actions, or one, or none.
The figure above shows the Configure Summary Table: Actions region, with options checked for the Show Create, Show Edit, and Show Delete options on the Action menu. It also shows a custom button and a custom action.

After you define an action, you can then expose it as a button or an Actions menu option in a variety of locations:

- On simplified pages
- Summary table on the overview page
- Default summary on the details page
- Summary table on a details page’s subtab
- Summary table on a tree node page for a child object
- Revenue table on the details page for the opportunity object

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.
After you define a link for an object, you can add that link to 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

Related Topics

- Defining Pages: Explained
- Creating a Work Area: Explained

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 objects listed below. 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>%253B<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:
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 below 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?
fnfGlobalItemId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM_CARD
&fnfTaskItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM&fnf=%3BsubTabName%253DOverview%253BAccountPartyId%253D123456%253B%3B%3Bfalse%3B256%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 `&fnf=%3BsubTabName%253DProfile`.

- Overview
- Profile
- SalesAccountTeam
- Contacts
- Assets
- Opportunities
- Quotes
- Leads
- Relationships
- Recommendations
- Notes
- Activities
- Conversations

- Activity.

For this object, use this direct page link URL pattern to open a list of all activities:

```
https://<hostname>:<port>/<application>/faces/FuseOverview?
fnfGlobalItemId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM_CARD&fnfTaskItemNodeId=ZCM_CUSTOMERCTRINFRA360_CUSTOMERS_CRM&fnf=%3BsubTabName%253DOverview%253BAccountPartyId%253D%253B%3Bfalse%3B256%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?`
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:

```
https://<hostname>:<port>/<application>/faces/FuseOverview?
fndGlobalItemNodeType=MKL_LEADS_CARD&fndTaskItemNodeType=MKL_LEADS&fnd=%3BsubTabName%253DSUMMARY%253DLeadId%253D<recordID>, such as recordID, such as 123456%253B%3B%3B%3B%3B%3B%3B%3B%3B%3B
```

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:
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%253DQuotes`.

- 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%253DObjectId%253D<YYYY>%253D%3B%3B%3Bfalse%253B%3B%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**

<table>
<thead>
<tr>
<th>Subtab</th>
<th>TZ0205P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtab Component ID</td>
<td>ExtGen_P1ToP2_1423192131410</td>
</tr>
</tbody>
</table>

Data Object TZ0205P2

Type Related object

P1ToP2FuseTab ObjectList View

<table>
<thead>
<tr>
<th>Label</th>
<th>TZ0205P2 Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descr...</td>
<td>RecordName</td>
</tr>
</tbody>
</table>

- **Partner**
  
  For this page, use this direct page link URL pattern:

  ```
  https://<hostname>:<port>/application/faces/FuseOverview?
  ZPM_PARTNERS_CARD&findTaskItemNodeId=ZPM_PARTNERS&findpartyId=253D100000151014782%3B3B%3Bfalse%3B256%3B%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%3B253%3B%3B%3Bfalse%3B256%3B%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%3B253%3B%3B%3Bfalse%3B256%3B%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:

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:
  

- **Customer.**
  
  For this object, use this direct page link URL pattern:
  

- **Lead.**
  
  For this object, use this direct page link URL pattern:
  
  http://<hostname>:<port>/crmCommon/faces/deeplink?ObjType=Lead&ObjId=99999700000

- **Opportunity.**
  
  For this object, use this direct page link URL pattern:
  
  http://<hostname>:<port>/crmCommon/faces/deeplink?ObjType=Opty&ObjId=99123456
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.
To enable the import and export of custom object data:

1. Confirm that you’re not in a sandbox.
2. In Application Composer, select an application on the main Overview page.
3. Select the **Import and Export** link in the Common Setup pane, or in the local area of the main Overview page.
4. 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 customize objects in Application Composer?**

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 customizations) and the Custom Objects Administration job role and Sales Administrator job role (for testing the customizations 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 customize. In Sales Cloud, Direct Selection mode is available when you customize 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, selectable UI components become apparent when you move your cursor over the UI component. Selectable UI components are highlighted and can be edited.

The following table describes how you can use each mode of Page Composer to customize 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 customize 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>
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. With Application Composer, administrators can make customizations at the site level only.

<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>Make field read-only or updatable</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Reorder fields in a Form</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Reorder table columns</td>
<td>Yes</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Hide or show table columns</td>
<td>Yes</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Set table column width with the mouse</td>
<td>Yes</td>
<td>No</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Set table column width and min width in percent or pixels</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
<tr>
<td>Make column sortable or not</td>
<td>No</td>
<td>Yes</td>
<td>Transactional pages (all non-dashboard pages)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customization 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 customizations 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>Customize 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 customizations (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>Select the MDS layer where you want to author customizations, such as at the site layer or job role layer</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Customization Task</td>
<td>Available in Page Composer (site, job role, external or internal level)?</td>
<td>Available in Application Composer (site level only)?</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>View results of customizations immediately</td>
<td>Yes, in the Page Composer design interface</td>
<td>Yes, in the Sales Cloud application that you are customizing</td>
</tr>
</tbody>
</table>

**Related Topics**
- Customizing Oracle Sales Cloud Pages Using Page Composer: Explained

**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
5 Extending Simplified Pages

Overview

Customize 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 customize simplified pages using Application Composer
- How to customize 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 customize 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 customize the simplified page regions that are extensible.

Related Topics

- Extending Simplified Pages for Accounts: Explained
- Extending Simplified Pages for Leads: Explained
- Extending Simplified Pages for Opportunities: Explained

Customizing 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 customize existing pages for standard objects. You can customize 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’ll learn which objects have simplified pages that are extensible, and how to customize 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 customize.

Objects whose simplified pages are extensible have a tab called Simplified Pages when the Pages node is selected. You can customize the pages listed under this Simplified Pages tab.

For example, the following standard objects have a set of simplified pages which you can customize:

- Account
- Activity
- Asset
- Contact
- Customer Contact Profile
- Forecast Territory Details
- Household
- Lead
- Note
- Opportunity
- Partner
- Program Enrollments
- Relationship
- Resource
- Sales Campaign
- Sales Lead

Customize Content
Use either Application Composer or Page Composer to customize 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 customizations that are role-based
- Hide or show fields
- Change field labels
- Reorder subtabs

Customizing Simplified Pages Using Application Composer

To customize simplified pages using Application Composer:

1. Select an application on the main Overview page.
2. In the object tree, select an object that offers a set of simplified 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 customize 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.

Customizing Simplified Pages Using Page Composer: Procedure

On a simplified page, you can customize 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 customizing 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 customizations.

**Note:** Any changes you make apply:

- Only to the page you’re on.
- To all or specific groups of users, depending on the customization layer you select before making changes.
Prerequisites
Activate a sandbox.

Customizing a UI Component
To update component properties:

1. Click your user name in the global area and select **Customize Pages**.
2. Select a customization layer, for example to make changes only for users with a specific job role.

> **Note:** When you customize a UI component for a specific job role, that job role must be assigned to you for you to test the customization 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 customize. You can tell you’re in this view when the Design button above 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 above 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 customization 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 customize 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 customizations 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 Customization Layers: Examples
- Setting Up Sandboxes: Procedure
- Customizing the Navigator and Springboard: Overview
- Role Provisioning and Deprovisioning: Explained

### 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.

See Working With Page Layouts: Explained.

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.

Note: To learn how to configure custom object Search and Select dialog boxes that users launch from desktop pages, see Defining Pages: Explained.

### Search and Select Dialog Boxes

A Search and Select dialog lets your end users search for and select object records at run time. 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 drop-down field or 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.
   
   See "Creating a Set of Simplified Pages for Custom Objects: Explained."

2. Click the Edit Picker link, and then 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 can then 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. This dynamic choice list field actually enables a one-to-many relationship between the Solution and Service Request objects.

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 run time, your users can 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 run time, 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 these objects listed below:

- 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 "Customizing 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**
  - 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.

- **Role**
  - 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.

- **Expression**
  - 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.
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 customizable header.

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.
  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.
- 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.

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 customized. 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 customize, 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.

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.
d. Click **Next**.

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:

```java
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:

```java
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.

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 customize 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. Select the Sales application.
3. Under the Objects navigation tree, expand Standard Objects, then expand the Opportunity object.
4. Click the Pages node.
5. Click the Simplified Pages tab.
6. 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.
7. 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
- Web content subtab

These subtab types are described in “Subtab Types: Explained”. Read that topic to learn how to add each specific type of subtab.

In this example, select the option for **Web content** and click **Next**.

8. In the Display Label field, enter **Customer Map Location**.
9. Under URL Definition, click the Expand Show Palette button, if it’s not already expanded.
10. Enter this groovy script:

```groovy
def myfinalURL = "http://www.bing.com/maps/?v2&where1="
def myURL1 = CustomerAddress_c != null ? CustomerAddress_c : ''
myfinalURL += myURL1
return(myfinalURL)
```

11. (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.

12. Click **Save and Close**, and then click **Done**.

13. When you navigate to the Edit Opportunity page at run time, you'll see that a new subtab, Customer Map Location, is now available.

If your end users entered an address in the Customer Address field while editing an opportunity record, then the map Web site should appear on the subtab, displaying the location of the customer’s address.

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, you’ll see that the Appointments subtab is no longer available.

**Related Topics**

- Adding an External Web Content Subtab to a Simplified Page

**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
- Web 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 below.

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.
- 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.

    
    \textbf{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.

   ![Application Composer](image)

   **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 customize this UI, select the Note object under the **Common** application, 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**.

### Web Content Subtabs

A Web content subtab exposes an external Web site right on an object’s details page. The Web 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.

To add a Web 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 **Web content**.

3. On the Web Content subtab configuration page, enter the display label for the subtab, and then define the URL to retrieve the subtab’s Web content.
   
   Optionally use the expression builder to build the URL expression that you need.

4. (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.

5. Click **Save and Close**.

The expression you build should include the following:

- Use the HTTP protocol.
- Optionally include field values from the current object as parameters, or user variables.
- Enclose static strings in double quotation marks.
  
  For example, "http://www.abc.com/".

For example:

```java
def myfinalURL = "http://www.bing.com/maps/?v2&where="
def myURL1 = ContactAddress_c != null ? ContactAddress_c : ''
def myURL2 = ContactCity_c != null ? ContactCity_c : ''
def myURL3 = ContactState_c != null ? ContactState_c : ''
myfinalURL += adf.util.GlobalEncodeField(myURL1 + " + myURL2 + " +myURL3)
```
return(myfinalURL)

Note: Before end users can view the contents of your web 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’ll 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. Select the Sales application.
3. Under the Standard Objects tree, expand the Sales Lead object.
4. Click the Pages node.
5. Select the Simplified Pages tab.
6. 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.
7. On the Edit Layout page, select the plus icon in the subtabs region.
8. On the Create Subtab page, select the Related object option.
9. Click Next.
10. Specify the values for the fields as follows:

<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>

11. 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

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 Leads page to an Edit Opportunity page, and then from the Edit Opportunity Page to 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 related topic Defining Pages: Explained.
When configuring a details page, as you move the fields from the Available Fields box to the Selected Fields box, the fields related to the object for which you are customizing 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) using which you 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 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 the drill down column and fields.

The 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 the 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 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. On the Overview page of the Application Composer, select Sales from the Application list.
4. Expand Standard Objects and click Sales Lead.
5. Click the Actions and Links link.
6. Click the Simplified Pages tab.
7. Edit the desired custom layout in the Details Page Layouts region.
8. Select the Summary tab, and click the edit icon.
9. 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 Action Links to Simplified Pages: Worked Example

This worked example illustrates how to create and add custom action links to simplified pages.

This example covers:

1. Creating a custom link that launches Google homepage.
2. Adding the custom link to the Leads page in the Simplified UI.
3. Verifying that the link appears on the Simplified UI.
Creating a Custom Link

In this step, you are creating an action link called Test.

<i>Note:</i> You must have a sandbox active before you begin your customization tasks.

1. Navigate to Application Composer.
2. In the left pane, select <b>Sales</b> from the Application drop down.
3. Expand <b>Standard Objects</b>, and then expand <b>Sales Lead</b>.
4. Click <b>Actions and Links</b>.
5. In the Sales Lead: Actions and Links page, select <b>Actions - Create</b>
6. Enter or select the following:

<table>
<thead>
<tr>
<th>Field or Region</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Label field</td>
<td>Enter <strong>Test</strong></td>
</tr>
<tr>
<td></td>
<td>After you enter Test, the <strong>Name</strong> field Automatically displays Test.</td>
</tr>
<tr>
<td>Type 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>URL Definition region</td>
<td>Enter &lt;&gt;'<a href="http://www.google.com">http://www.google.com</a>'&lt;/'&gt; (along with single quotes)</td>
</tr>
</tbody>
</table>

<i>Note:</i> In your script that builds the URL, do not use create, update, or delete statements that manipulate another object’s data. Your script should only read data and then generate the URL.
This figure shows the Create Action or Link page for the Sales Leads object.

7. In the script region, enter ‘http://www.google.com’.
8. Click the Validate icon. A success message appears.
This figure shows the URL definition region and the validation message.

9. Click the **Save** button to save the new link.

   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.
6. Click Edit icon in the Summary region. The Edit Lead page opens.

7. In the Available Fields column, locate Test - Link field and move it to the Selected Fields column.

Testing the Link in Simplified Pages

In this step, you verify the addition of the field.
To test whether the Test link you created appears in Simplified Pages:

1. Open the Simplified Pages for Leads.
2. Edit a lead and note the newly created Test link in the Edit Lead: <Name> Summary page.

This figure shows the Test link.

3. Click the Test link. The Google homepage appears.

Customizing Work Area Lists

Creating Custom Work Area Lists

Watch: This tutorial shows how sales administrators can create custom work area lists (saved searches) for specific job roles or for the sales organization as a whole.

Customizing Work Area Lists: Deleting

Watch: This tutorial shows how sales administrators can delete custom lists and hide lists provided by Oracle in the different Oracle Sales Cloud work areas.

Procedures

Sales administrators can use the procedures in this topic to customize 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.
To customize work area lists, you do the following:

1. Navigate to the work area and get ready to customize by activating a sandbox and opening Page Composer.
2. Make your customizations. You can:
   - Create a custom search.
   - Hide searches provided by Oracle.
   - Delete any custom searches you created.
3. Publish the sandbox to make your customizations available to users.

### Getting Ready to Customize Lists

Follow these steps to enable a sandbox and get ready to start customizing lists:

1. Navigate to the work area landing page you want to customize.
2. Click your user name at the top right corner of the page, 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.
3. From the same user name menu, select **Customize Pages**.

   The Customize Pages window appears.
4. Select one of the two customization layers:
   - **Site** to make the customization available to everyone.
   - **Job Role** to target a specific job role you select from the list.

   The Editing: User Interface Page Composer toolbar appears at the top of the page with the **Design** mode selected.

   You are now ready to create the customized saved search. After you complete your customizations and test them, you must publish the sandbox to make your customizations available to your users.

### Creating a Custom Saved Search

With both the sandbox and the Editing: User Interface Page Composer toolbar displayed on the top of the page, follow these steps to create a custom saved search. You create a new saved search by editing an existing list and saving it under a new name.

1. In the work area landing pad, click **Show Advanced Search** (the filter icon) to the right of the **List** field.

   The Saved Searches window appears.
2. From the **Saved Search** list, select a saved search to use as the starting point for creating a new one.

   **Tip:** If you want to create a saved search using only one field, such as a custom field, then select one of two predefined saved searches: Name and Close Date. 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 to the search criteria. 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 a custom field to your search, then the field must be indexed for best search performance.

   - Select different operators and values for the fields.
     You must enter values for fields marked with an asterisk.
   - Reorder the filter conditions by clicking **Reorder**.
     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. If 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 Saved Searches window

**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** (the filter icon) to the right of the **List** field.
   The Saved Searches window 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 in he 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 navgate to the work area.

1. In the work area landing pad, click Show Advanced Search (the filter icon) to the right of the List field.

   The Saved Searches window appears.

2. From the Saved Search list, select a saved search you want to delete.


   The Personalize Saved Searches window appears. The Saves 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

FAQs 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.
How can I post announcements on the home page?

Use the Announcements page to create, edit, and delete announcements. From the Navigator menu, select **Tools - Announcements**.

- Only the announcement’s content (not subject) appears on the home page.
- If social networking features appear instead of announcements, then on the Set System Options page, change the home panel settings to display announcements. To open this page from the Navigator menu, select **Tools - Structure**, and then click **Set System Options**.
- What you do on the Announcements page applies immediately to all users, even if you saved your changes while a sandbox is active and not yet published.

**Related Topics**

- Setting Up Sandboxes: Procedure
- Defining Settings for Home and Navigation: Explained
6 Extending Desktop Pages

Defining Pages: Explained

After you extend an Oracle Sales Cloud application’s object model using Application Composer, your next step is to expose those new objects and fields to users. To expose new objects and fields to users on desktop pages, navigate to the object’s Pages node and select the Desktop Pages tab. This tab lets you create new pages and customize existing pages. Customizing and creating Sales Cloud desktop pages is a simplified process because the pages available to display an object and its fields are limited to a set of page types: every top-level Sales Cloud object has an overview page, a creation page, and a details page, collectively known as a work area.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

After you create a custom object, you can create its corresponding work area by using the work area wizard, available from the Desktop Pages tab. After a work area is created, the Desktop Pages tab provides links to work area configuration pages, which you can use to add or remove fields for display. This combination of page types, configuration pages, a wizard-driven page creation process, and the ability to add links to the Navigator menu means that you can quickly and easily expose custom object model extensions to your users.

You can expose new fields in the alternative set of simplified user interface pages, if available, using the Simplified Pages tab. Note that you configure the look and feel of simplified pages using page layouts. For more information on page layouts, see the Extending Simplified Pages chapter.

You can also expose new objects and fields in mobile applications using the Mobile Pages tab.

Before you begin to customize or create new pages for a Sales Cloud application, review the following aspects of the Application Composer desktop page creation process:

- Understanding Page Types
- Using the Pages Overview Page
- Defining Pages
- Creating a Work Area
- Object Security on Pages
- Using Page Composer to Customize Pages

Understanding Page Types

Every top-level Sales Cloud object can be displayed on a set of standard page types or in the regional area: the regional search in the regional area, an overview page, a creation page, and a details page.

- Regional Area
  
The regional area is located on the left side of the page and includes a search pane. You can customize the regional search pane by specifying the fields to display.
- Overview page
The overview page provides a list of records for an object and is the starting point in a Sales Cloud application for users to view and manage data. This page is where you can search for existing records and create new records. Users access an object’s overview page from the Navigator menu at run time.

Only top-level objects have an overview page. If an object was created as a child to another object, then the child does not have an overview page.

The overview page includes two regions, each of which has its own configuration page:

- **Local search region**
  
  The local search region is displayed above the summary table. Users can enter search criteria to refine the list of records in the summary table and then save this list as a saved search in the local search region.

- **Summary table**
  
  The summary table includes a list of the object records. Depending on the security setup, users can create a record, delete a record, or drill down into an existing record.

  Optionally, define saved searches that users can select at run time to constrain the list of records displayed in the summary table.

  To create new saved searches for your users, navigate to the run time search page where you want to create your saved search, and access Page Composer using the Administration menu. Then, enter and execute your search, and save it using the Save button under the search criteria in the application. You can also edit the saved searches that were created in Application Composer in earlier releases, by navigating to the Saved Search node for a custom or standard object.

- **Creation page**
  
  The creation page is the page where users can create new records for an object. Depending on the security setup, users access the creation page by clicking the New icon or by selecting the New menu item from the Actions menu on the summary table’s toolbar.

- **Details page**
  
  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 include both a default summary and a detailed summary. The default summary includes the primary object fields and is always displayed to users. The detailed summary includes additional fields for an object. You cannot add the same field to both the default and detailed summaries.

  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.

  **Note:** Some Sales Cloud objects, known as common objects, do not have a standard work area. These include the note common component and common objects (resource, customer contact profile, account, address).
Using the Pages Overview Page

The Pages overview page in Application Composer provides an overview of the set of standard pages for an object.

To access the Pages overview page:

1. Select an application on the main Overview page.
2. Select a standard or custom object in the object tree and expand it.
3. Select the **Pages** node.

Only top-level objects have pages that you can configure. A child object does not exist outside the context of the parent object, and does not have its own work area.

From the Pages overview page, you can:

- Create a new set of pages for an object, collectively known as a work area, if no set of pages has been created yet.
- View the pages where the object is already exposed to users, and further customize those pages by adding or removing fields.
- Create one or more subtabs to display on the object’s Details page.
- Create mobile application pages in Oracle Fusion Mobile Sales by using a wizard.
Similar to the work area creation process, the creation process for mobile pages uses a wizard where you can select which custom fields and related objects to add to mobile pages. Select the Mobile Pages tab to access the mobile pages wizard.

- Expose new fields in an alternative set of simplified user interface pages, if available for a standard object, by selecting the Simplified Pages tab.

Note that the Simplified Pages tab displays only if a set of simplified pages exists for the standard object.

**Defining Pages**

Use configuration pages to specify which fields are displayed on the standard application pages for an object. After you create new objects and fields, navigate to the Pages overview page. The Pages overview page contains hyperlinks to the configuration pages for an object’s existing work area. Use these configuration pages to customize the object’s work area pages, for example add newly created fields to a creation page.

**Note:** If the Pages overview page does not contain these configuration page hyperlinks, then the object does not yet have a work area, and you must create one if you want the object to be visible to users at run time. If a standard object does not expose the same set of configuration page links that you see for a custom object, it is because the standard object supports additional page types unique to that object.

Use the configuration pages available from the Pages overview page as follows:

- Navigator Menu
For custom objects, specify the display label that appears in the Navigator menu at run time. For standard objects, use the Manage Menu Customizations task in the Setup and Maintenance work area to change the display label for standard object work areas.

- Regional Pane
  Select which panes to display in the regional area. You can also specify if you want the regional area and individual panes to be expanded (or collapsed) by default.

- Overview Page
  The overview page includes two regions, the local search and the summary table, and includes configuration pages for each.
    - Edit Local Search
The following figure show the Edit Local Search region, one of the work area configuration pages in Application Composer.

i. Select the fields that will be used to search in the local search region.

Join fields are not available for selection in the local search region; 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.

The list of fields available for selection is displayed to you in a single column, although the creation page is formatted as a page with three columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you select is displayed in the third column, the fourth field you select is displayed in the first column again, and so on.

During field creation, consider indexing any fields that you plan to display as search criteria for your custom objects.
The following figure shows the Edit Local Search field operators which you can select as part of your configuration of the local search region.

<table>
<thead>
<tr>
<th>Selected Field</th>
<th>Default Operator</th>
<th>Required</th>
<th>At Least One is Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatedBy</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreationDate</td>
<td>Equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CurrencyCode</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastUpdateDate</td>
<td>Equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastUpdatedBy</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales Name</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ii. For any field, you can select the following options:

- Required: The user must include this field in a search at run time.
- At Least One is Required: The user must include at least one of the selected fields in a search at run time.
- Default Operator: The user can define each search field value by using one of the listed operator options. For example, you can specify that CreationDate is equal to or occurs before or after the date the user enters, or that Sales Name starts with a specific letter.
Edit Summary Table

- **Available Fields**
  - AccountNumber
  - AccountNumber
  - Address
  - AlertsExistFlag
  - AsgnExceptionFlag
  - AttachmentEntityName
  - BdgtAmtCurcyCd
  - BestCase
  - BudgetAmount
  - Budgeted
  - BudgetedFlagLowSwitcher
  - BusinessUnit

- **Selected Fields**
  - Group of Fields 1
  - Revenue
  - Currency
  - Group of Fields 2
  - PrimaryContact
  - CloseDate
  - SalesStage
  - WinProbability

**i.** Select the fields that you want to display as columns in the summary table.

**ii.** Add custom actions or links to the summary table, if you previously created them.
Creation Page

Application Composer

Edit Creation Page

Configure Creation Page

Available Fields
- Account Number
- Account Number
- Address
- AlertsExtFlag
- AsgnExceptionFlag
- AttachmentEntityName
- BdgtAmtCurcyCd
- Best Case
- Budget Amount
- Budgeted
- BudgetedFlag

Selected Fields
- Opportunity Name
- ContactName
- SalesChannel
- Currency
- Revenue
- Win Probability
- Close Date
- SalesMethod
- SalesStage

The list of fields available for selection is displayed to you in a single column, although the creation page is formatted as a page with three columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you select is displayed in the third column, the fourth field you select is displayed in the first column again, and so on.
Details Page

Application Composer

Edit Details Page Summary Form

Object Name  Opportunity

Configure Default Summary

Available Fields
- Account Number
- Account Number
- Address
- AlertsExistFlag
- AsgnExceptionFlag
- AttachmentEntityName
- BdgtAmntCurrcyCd
- Budget Amount
- Budgeted
- BudgetedFlag
- DRVSwitcher

Selected Fields
- Opportunity Name
- Group of Fields 1
- Owner
- Revenue
- Worst Case
- Best Case
- Currency
- Status
- Win/Loss Reason
- Close Date
- Sales Method

Configure Detailed Summary

Available Fields
- Account Number
- Account Number

Selected Fields
- Customer Account
- Opportunity Number

a. Select the fields that you want to display on the object’s details page, including both the default summary and detailed summary regions. Include the primary object fields in the default summary, since the detailed summary could be collapsed when users navigate to this page.

The list of fields available for selection is displayed to you in a single column, although the details page is formatted as a page with three columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you select is displayed in the third column, the fourth field you select is displayed in the first column again, and so on.

b. Add custom buttons, links, and actions to the details page, if you previously created them.

c. If you want to allow attachments, add the Attachments field to the Selected Fields list in the Configure Detailed Summary section.

d. The Pages Overview page also lets you configure subtabs that display on the details page. Subtabs include information that is related to the object record. Adding subtabs to a details page is discussed in a related topic.

If the object uses a tree rather than subtabs to display related pages, then you can configure tree nodes that you add to the object’s tree. The information displayed in both subtabs and trees can be derived from
other objects or from a source outside Oracle Sales Cloud. For more information on adding tree nodes to an object’s tree, see the related topic on this subject.

- Reusable Regions

At run time, users can launch the Select and Add dialog box from a dynamic choice list field to search for and select a value. Users can use the dynamic choice list field at run time, and can launch the picker to limit their choices and more easily find and make the right selection. Use the Create Picker link or Edit Picker link to access the Search and Select picker dialog box for an object, which you can start from any dynamic choice list that is based on that object.

Note: You can’t use Page Composer to customize the Search region inside Search and Select dialog.

In the Search and Select picker dialog box you can specify which fields you want to appear in the search region and in the search results table. If you create a dynamic choice list that is based on a custom object, you must create the picker for the same custom object.

To create or edit a picker, use the following steps.

a. Create or edit the dynamic choice list field for which you want to create a picker. In the Related Object field for that dynamic choice list, make sure that you have selected the correct object on which you want to base the dynamic choice list.

b. On the Pages overview page, click Create Picker or Edit Picker.

The Edit Picker page opens.

c. In the Configure Picker Search list, select the fields that you want to appear in the search region.

d. In the Configure Picker Table list, select the fields that you want to appear in the search results table.

At run time, the user can select a value from the dynamic choice list field by using the Search and Select picker to find the right value among all the other values.

a. Navigate to the object’s work area for which you created the picker and dynamic choice list.

b. In the Overview page, click a record to drill down to the details page.

The following figure shows the details page for a custom object named Primary with a dynamic choice list field named Opportunity List.

c. Click the magnifying glass icon next to the dynamic choice list field to open the Search and Select picker dialog box.
The user can add the selected value in the dynamic choice list field by using the following steps:

a. In the Search and Select picker dialog box, enter the search criteria for the field.

b. Click **Search**.

c. Select the record.

d. Click **OK**.

The value is populated into the dynamic choice list field.

- Edit Regional Search
Select the fields the user can search on using the search pane in the regional area.

**Edit Regional Search**

Object Name: Sales Tool

**Edit Regional Search**

Select the fields to appear in the regional search.

**Available Fields**

- Last Update Date
- Parameter

**Selected Fields**

- Created By
- Creation Date
- Currency Code
- Last Updated By
- Sales Tool Name
- Value

<table>
<thead>
<tr>
<th>Selected Field</th>
<th>Default Operator</th>
<th>Required</th>
<th>At Least One is Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created By</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creation Date</td>
<td>Equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Currency Code</td>
<td>Equal to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Updated By</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales Tool Name</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>Starts with</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To display the entire regional search pane, check the Enable check box. Otherwise, the regional search shows only the selected fields.

For any field, you can select the following options:

- **Required**: The user must include this field in a search at run time.
- **At Least One is Required**: The user must include at least one of the selected fields in a search at run time.
- **Default Operator**: The user can define each search field value by using one of the listed operator options. For example, you can specify that Creation Date is equal to or occurs before or after the date the user enters, or that Sales Name starts with a specific letter.

Because some Sales Cloud objects (common components and common objects) do not have a standard work area, the configuration pages available from their Pages overview page are different than described previously. For example, the common address object has configuration pages for customizing the overview page, creation page, and address details form.

The common account object has configuration pages for customizing only the details form and create form.

When you customize pages for common objects, the changes you make are reflected across the multiple applications where the object is used, provided that the applications also share the same metadata repository.

**Creating a Work Area**

When first created, top-level custom objects do not yet have pages in a run time Sales Cloud application where those objects are exposed to users. For each custom object that you create, you must create the set of pages where the records that belong to the object will be exposed to users.

Application Composer uses a wizard to walk you through the creation of these pages, collectively known as a work area. For more information on creating a work area, see the related topic on this subject.

Do not create a work area for child objects.
Object Security on Pages

After you create custom objects and fields, you then expose them on pages for your users. By default, the object and its records are visible and can be edited only by a user who has the Custom Objects Administration job role. Grant additional access manually for either an object or role, using Application Composer’s security policy configuration pages.

The security options available to you for restricting access to custom objects, including child objects, are discussed in a related topic.

Using Page Composer to Customize Pages

After you create a set of new pages, or edit preexisting pages delivered by a Sales Cloud application, you should avoid using Page Composer to edit those pages again.

Related Topics

- Oracle Sales Cloud Mobile Extensibility: Explained
- Making Custom Object Pages Visible to Users: Explained
- Customizing Oracle Sales Cloud Simplified Pages: Explained

Creating a Work Area: Explained

This topic describes how to create a work area to create a new work area in order to create a custom object. When custom objects are first created they don’t yet have pages in a run time Sales Cloud application where those objects are exposed to users. After you create a top-level custom object, you must create a set of user interface pages for that object. Depending on your implementation, you can create either a set of desktop pages or a set of simplified pages for the custom object. Application Composer employs a wizard approach to walk you through the creation of these pages. After you create the initial set of pages, either desktop or simplified, you can always continue to customize those pages. You do not create user interface pages for child objects.

Review these aspects of the work area creation process in Application Composer before you create a new work area for a custom object:

- Using the work area wizard
- Configuring the Navigator menu
- Configuring the local search region
- Configuring the overview and creation pages
- Configuring the details page

Note: You can customize desktop pages if you have implemented Oracle Sales Cloud in Release 9 or earlier. For all implementations, using simplified pages is highly recommended.

To create and modify pages displayed on a mobile device, use the separate Mobile Pages wizard which is also available from the object's Pages Overview page.
Accessing the Work Area Wizard

Access the wizard on the Pages Overview page using the same navigation path that you use to configure pages in an existing work area. However, if a work area hasn't yet been created for an object, then hyperlinks to the work area configuration pages aren't displayed. Instead, the Pages Overview page displays only a single hyperlink to start the work area wizard.

To access the work area wizard:

1. Select an application on the main Overview page.
2. Select a standard or custom object in the object tree.
3. Select the Pages node.

Only top-level objects have pages that you can configure. A child object doesn't exist outside the context of the parent object, and does not have its own work area.

4. Select the hyperlink to start the work area wizard.

✓ Note: Use the work area wizard to create a work area. Use the work area configuration pages to customize existing work area pages.

Exploring the Navigator Menu

As part of the work area creation for a custom object, you must specify the object label that appears in the Navigator menu at run time. The label you specify is what users will select to navigate to this work area.

To explore the Navigator menu:

- Select a menu category under which the object label appears.
- Adjust the position of Navigator menu items within the selected menu category.

For example, move your newly created object label to appear at the top of the list.

After creating the work area for a custom object, the work area label automatically appears in the Navigator menu without you having to reauthenticate.

Changing the object label on the Navigator menu is available in Application Composer only for custom objects. For standard objects, use the Manage Menu Customizations task in the Setup and Maintenance work area to change the display label for standard object work areas.

Regional Search Parameters

Extending the Regional search is part of the first step in creating a work area. By adding custom parameters, you can expand the search parameters in the Regional area for both custom and standard objects in a Sales Cloud application at run time.

Each field can have one of the following properties:

- Required: Sets a field as required on the object search region. This field must be populated.
- At least one is required: Sets at least one field as a required field or a collection of required fields. You must populate one field from this group.
- Default operator: Sets an operator field that appears in the search as the default.
Local Search Region

In the configure search step, select the fields that you want to display as search criteria fields in the local search region. Adding fields to this region is optional.

The list of fields available for selection is displayed in a single column, although the local search region is formatted as a region with two columns.

You can’t control the order in which the chosen fields are displayed in the local search region.

During field creation, consider indexing any fields that plan to display as search criteria for your custom objects.

Configuring the Overview and Creation Pages

This section describes how to configure the overview and creation page using Application Composer.

To configure the overview page:

1. Select the fields that you want to display as columns in the summary table, on the object’s overview page.
2. Select the drill-down column for the summary table.
   - Objects appearing under the drill-down column appear as links on the summary table. Clicking a link takes you to that object’s details page. You can’t change the drill down column after the work area has been created.
   - **Note:** You’re only specifying which column displays the link in the summary table. Typically, you specify the object’s name field or the field that you use for identifying a record.
3. Add custom actions and links to the summary table, if you previously created them.
4. Select the fields that you want to display on the object’s creation page.
   - The fields that you select must include the object’s required fields.
   - The list of fields available for selection is displayed to you in a single column, although the creation page is formatted as a page with three columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you select is displayed in the third column, the fourth field you select is displayed in the first column again, and so on.

Configuring the Details Page

This section describes how to configure the details page. From within the details page you can also add subtabs, custom buttons, links, and actions.

A details page can have subtabs, which include information that’s related to the object record. For example, the details page for an opportunity could include a subtab that lists customer contacts or previous orders.

1. Select the fields that you want to display on the object’s details page, including both the default summary and detailed summary regions. Include the primary object fields in the default summary because since the detailed summary might be collapsed when users navigate to this page at run time.
   - The list of fields available for selection is displayed to you in a single column, although the creation page is formatted as a page with three columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you select is displayed in the third column, the fourth field you select is displayed in the first column again, and so on.
2. Add custom buttons, links, and actions to the details page, if you previously created them.

3. To add subtabs to a details page, create the work area first, then navigate back to the Pages Overview page. Adding subtabs to a details pages is discussed in a related topic.

4. If you want to allow attachments, then add the Attachments field to the Selected Fields list in the Configure Detailed Summary section.

**Related Topics**

- Attachments: Explained
- Oracle Sales Cloud Mobile Extensibility: Explained

### Subtabs: Explained

Every top-level Oracle Sales Cloud object has a details page, also known as the edit page, as part of its work area. This is the page where users can view and edit details about a particular record. You can customize any object’s details page by adding 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.

Review these aspects of the subtab creation process in Application Composer before you begin to add subtabs to an object's details page:

- Using the Details page
- Adding subtabs to simplified pages
- Adding subtabs to desktop pages

Only top-level objects have details pages that you can configure. A child object does not have its own work area.

Subtabs and tree nodes are two master/detail UI patterns which Oracle Sales Cloud applications support. For custom objects, only subtabs (not tree nodes) are supported.

If a standard object, such as the Partner object, uses a tree to display its related pages, then you would expose child or related objects as tree nodes instead of subtabs on a details page. Adding tree nodes is discussed in a related topic.

### Using the Details Page

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.

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.

Adding Subtabs to Simplified Pages
Add a subtab to a simplified details page from the Simplified Pages tab, Details Page Layouts region.
To add a subtab to a simplified details page, see "Adding and Hiding Subtabs Using Application Composer: Explained".

Adding Subtabs to Desktop Pages
Add a subtab to an object’s details page from that object’s Pages Overview page. The details page must exist already; you cannot add subtabs when first creating a work area.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To add a subtab to a desktop details page:

1. Select an application on the main Overview page.
2. Select an object in the object tree.
3. Select the Pages node.
4. Click the Desktop Pages tab in the local area.
5. In the Details Page region under the Subtabs heading, click the Create icon to create one or more subtabs to display on the object’s details page.
6. Select the type of subtab you want to add:
   - Related object subtab
   - Child object subtab
   - Context link subtab
   - Notes subtab
   - Web content subtab
   These subtab types are described in Subtab Types: Explained. Read that topic to learn how to add each specific type of subtab.

Related Topics
- Subtab Types: Explained
- Object Relationships: Explained
- Adding and Hiding Subtabs Using Application Composer: Explained

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
- Web 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 below.

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 customize this UI, select the Note object under the Common application, 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.

Web Content Subtabs

A Web content subtab exposes an external Web site right on an object’s details page. The Web 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.

To add a Web 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 **Web content**.

3. On the Web Content subtab configuration page, enter the display label for the subtab, and then define the URL to retrieve the subtab’s Web content.

   Optionally use the expression builder to build the URL expression that you need.

4. (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.

5. Click **Save and Close**.

The expression you build should include the following:

- Use the HTTP protocol.
- Optionally include field values from the current object as parameters, or user variables.
- Enclose static strings in double quotation marks.

For example, "http://www.abc.com/".

For example:

```java
def myfinalURL = "http://www.bing.com/maps/?v2&where="
def myURL1 = ContactAddress_c != null ? ContactAddress_c : ''
def myURL2 = ContactCity_c != null ? ContactCity_c : ''
def myURL3 = ContactState_c != null ? ContactState_c : ''
myfinalURL += adf.util_GlobalEncodeField(myURL1 + " " + myURL2 + " " +myURL3)
```
return(myfinalURL)

> **Note:** Before end users can view the contents of your web 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**

- Adding and Hiding Subtabs Using Application Composer: Explained
- Adding Subtabs: Example
- Creating and Reordering Subtabs in Simplified Details Pages: Worked Example

**Tree Nodes: Explained**

Some Oracle Sales Cloud standard objects, such as the Partner object, use a tree to display its related desktop pages. When configuring an object's work area, you can optionally display details that are related to the current object by adding tree nodes to the object's tree, and specifying the source of tree node data. Tree node data can be derived from another object, or from another source outside the current Sales Cloud application altogether. Add a tree node to a standard object's tree from that object's Pages Overview page in Application Composer.

This is available for desktop pages only.

> **Note:** You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

Review these aspects of the tree node creation process in Application Composer before you begin to add tree nodes to an object’s tree:

- Adding tree nodes
- Tree node types:
  - Child tree nodes
  - Related object tree nodes
  - Context link tree nodes
  - Web content tree nodes

Subtabs and tree nodes are two master/detail UI patterns which Oracle Sales Cloud supports.

For custom objects, only subtabs (not tree nodes) are supported.

For standard objects that are already using tree nodes, such as the Partner object, additional details adopt the same tree node pattern. In other words, if a standard object uses a tree to display its related pages, then you would expose child or related objects, for example, as tree nodes instead of subtabs on a details page. Adding subtabs is discussed in a related topic.
Adding Tree Nodes

Add a tree node to an object’s tree from that object’s Pages Overview page.

To add a tree node to an object’s tree:

1. Select an application on the main Overview page.
2. Select a standard object, such as the Partner object, in the object tree.
3. Select the Desktop Pages node.

Only the top-level objects, Partner and Sales Account, let you add tree nodes.

However, the Sales Account object is available for customization 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, available in the Common application.

4. On the Pages Overview page, click the Create Tree Node icon to create one or more tree nodes to display on the object’s tree.
5. Select the type of tree node you want to add.

Child or Related Object Tree Nodes

A relationship is a foreign key association between two objects. Using Application Composer, you can create a one-to-many relationship or a many-to-many relationship between two objects. Once relationships are created, you can expose the "many" objects on a tree node that is displayed on the "one" object’s tree. For example, a partner can have multiple contacts associated to it. To enable your end users to associate a list of contacts with a specific partner, you must first create a one-to-many relationship between the partner and contact objects. Then, you can add the Contacts tree node to the partner’s tree.

Application Composer lets you add a tree node to an object’s tree for either a child object or related object.

To add a child or related object tree node to an existing tree:

1. On the Pages Overview page, click the Create Tree Node icon.
2. Select either Child object or Related object.
3. On the tree node configuration page:
   a. Select the tree node category and enter the tree node label.
   b. Select the object that is to be exposed on the tree node page.
   c. Set the position of the new tree node.
   d. Optionally hide the New and Delete buttons that appear on the tree node page at run time.

   For child object tree node pages, you can also optionally hide the Edit button.
   e. Select which fields and links you want to display on the tree node page’s summary table at run time.

   You can configure fields and links for the main summary table which lists the child object records or related object records.
   f. Select which buttons you want to display on the tree node page at run time.

   📝 Note: This region appears only if you previously created buttons for this object.
   g. Select which fields you want to display on the tree node page’s detail form at run time.
You can configure fields for the detail form that appears under the summary table. If the tree node’s object is a child object, then users can enter child object data into this detail form at run time. Always include required fields in this section.

If the tree node’s object is a related object, then users can associate an existing record of the tree node object to the master object of the page. However, to create new related object records, users must do so in the object’s own creation page.

Context Link Tree Nodes

A context link tree node page 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 tree node pages are read only.

To add a context link tree node to an object’s tree:

1. On the Pages Overview page, click the **Create Tree Node** icon.
2. Select **Context Link**.
3. On the Context Link tree node configuration page:
   a. Select the tree node category and enter the tree node label.
   b. Select the object that is to be exposed on the tree node page.
   c. Set the position of the new tree node.
   d. 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.

   e. Select which fields you want to display on the tree node page’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.
   f. Select which fields you want to display on the tree node page’s read-only detail form at run time.
      You can configure fields for the detail form that appears under the summary table.

Web Content Tree Nodes

A Web content tree node page exposes an external Web site. The Web content is a result of the expression that you define which builds the intended URL.

For example, on the Contact tree node page, let us assume that you want to add a map showing 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 and then pass the URL to the Bing Maps API.

To add a Web content tree node to an object’s tree:

1. On the Pages Overview page, click the **Create Tree Node** icon to create one or more tree nodes to display on the object’s tree.
2. Select **Web Content**.
3. On the Web Content tree node configuration page:
   
   a. Select the tree node category and enter the tree node label.
   b. Set the position of the new tree node.
   c. Define the URL to retrieve the tree node page’s Web content.

   Optionally use the expression builder to build the URL expression that you need.

The expression you build should include the following:

- Use the HTTP protocol.
- Optionally include field values from the current object as parameters, or user variables.
- Enclose static strings in double quotation marks.

For example, "http://www.abc.com/".

For example:

```groovy
def myfinalURL = "http://www.bing.com/maps/?v2&where1="
def myURL1 = ContactAddress_c != null ? ContactAddress_c : ''
def myURL2 = ContactCity_c != null ? ContactCity_c : ''
def myURL3 = ContactState_c != null ? ContactState_c : ''
myfinalURL += adf.util.GlobalEncodeField(myURL1 + " " + myURL2 + " " + myURL3)
return(myfinalURL)
```

**Related Topics**

- Object Relationships: Explained
- Groovy Scripting: Explained
- Groovy Scripting: Examples
7 Extending Mobile Pages

Overview

Read this chapter to learn how to extend 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 customizations for any particular device
- How to add an Oracle Business Intelligence report to your mobile application
- How to test mobile application customizations using a sandbox

Oracle CX Enterprise Mobile

Customizing Oracle CX Enterprise Mobile: Explained

You can customize the Oracle CX Enterprise Mobile (Enterprise Mobile) iPhone and Android applications using Application Composer. Using Application Composer, you can manage which objects and fields are visible on the Enterprise Mobile application without having to carry out specific customizations for a particular device.

Creating a Page Layout for a Feature

You can create a List, Detail, or Edit page layout for an Enterprise 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?

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 a Custom Object to a Page Layout
You can add custom objects to your Enterprise 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 a Custom Object to a Page Layout: Procedure topic for more details.

Testing Customizations
After you have customized Enterprise Mobile using Application Composer, you should test your customizations before distributing them to your user’s mobile devices. See the Testing Oracle CX Enterprise Mobile Customizations: 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 Enterprise Mobile (Enterprise Mobile) feature. Creating your own layout enables you to select the fields your users will see for a feature’s views in Enterprise Mobile.

Tip: You can also specify which user roles can view a particular layout, and you can create criteria that have to be met to display the layout. See the Adding a Role to a Layout: Worked Example topic for more details about adding user roles. See the Creating Criteria for a Feature Page Layout: Procedure topic for more information about adding advanced criteria.

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 customizations.
3. Open Application Composer by selecting Application Composer under the Tools 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, expand the subheader of the feature that you want to create a layout for.
7. 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 drag and drop 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 dragging 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 Enterprise Mobile Customizations: Worked Example topic for details about how to check your mobile customizations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

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 drag and drop 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 drag 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.

Restriction: 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 Enterprise Mobile (Enterprise 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.

Restriction: 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 Enterprise 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 customizations.
3. Open Application Composer by selecting Application Composer under the Tools 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, expand the subheader of the application feature that you want to add the report to. In this example, expand the Opportunities subheader.
7. Expand the subheader of the layout you want to add the role to. In this example, expand the Detail subheader.
8. Select the layout you want to add a role to. 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 Enterprise Mobile Customizations: Worked Example topic for details about how to check your mobile customizations.
2. When you are happy with your changes, publish your sandbox to distribute your customizations to all Enterprise Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

Related Topics
- Using Sandboxes: Explained

Creating Criteria for a Feature Page Layout: Procedure

This procedure shows you how to create criteria for an Oracle CX Enterprise 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.

- **Restriction:** 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 customizations.
3. Open Application Composer by selecting Application Composer under the Tools 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, expand the subheader of the relevant feature.
7. Expand the subheader of the relevant page view and select the page layout you want to add criteria to.
8. In the Advance Criteria pane, click Add.
9. Create your criterion by selecting a field, operator, and then entering the relevant field value.

- **Restriction:** You can’t select a field value from a list of values (LOV), so you will need to type in the value if you would normally select it from an LOV.

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 Enterprise Mobile Customizations: Worked Example topic for details about how to check your mobile customizations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

Adding a Custom Object to a Page Layout: Procedure

This procedure shows you how to add a custom object to your Oracle Sales Cloud Enterprise Mobile (Enterprise Mobile) application. Sales objects are also known as features in Enterprise Mobile, and when you create a custom Sales object in Application Composer a corresponding feature is also created in Enterprise 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 Enterprise 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 customizations.
3. Open Application Composer by selecting Application Composer under the Tools 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 Available Feature pane, click on the feature you want to add, and drag and drop it onto the mobile interface designer.
7. Click **OK** in the information message about the views that will be created.
8. 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.
9. If you want to add fields to the picker, click on the fields you want to add in the Available Fields pane, and drag and drop them onto the mobile interface designer.
10. When you have finished adding features, click **Save** in the top right-hand side of the Application Composer page.
11. Test and publish your new features. See the Testing Oracle CX Enterprise Mobile Customizations: Worked Example topic for details about how to check your mobile customizations. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

Adding Oracle Business Intelligence Reports to a Sales Object's Analytics Tab: Procedure

To view Oracle Business Intelligence Analysis (OBI) reports in a Sales object’s Analytics tab (for example, an Accounts or Contacts Analytics tab), you must first perform some configuration tasks from the web application.

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 and select the **Sales** application.
3. Select **Mobile Application Setup** under the Common Setup list.
4. In the Application Feature pane, expand the subheader of the sales object that you want to add the report to.

   **Restriction:** You can add reports to the following Sales objects only: Account, Opportunity, Contact, Lead, Partner, and Deal Registration.

5. Expand the Analytics subheader and select **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 OBI reports that you have available in your catalog.

7. Drag and drop 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 OBI report. To further filter the report, 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 OBI report parameter value so that only the data relevant to the Sales object you are viewing is displayed. 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. For a Leads report, you could filter using the $LeadId or $Name parameters.

> **Note:** You can only provide parameter values to filters that have already been defined in the original OBI report.

10. Click the filter’s tick 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 Enterprise Mobile Customizations: Worked Example topic for details about how to check your mobile customizations.
14. When you are happy with your changes, publish your sandbox to distribute your customizations to all Enterprise Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

**Related Topics**

- Using Sandboxes: Explained

### Testing Oracle CX Enterprise Mobile Customizations: Worked Example

After you’ve customized Oracle CX Enterprise Mobile using Application Composer, you should test your customizations before distributing them to your user’s mobile devices.

#### Task Summary

To test your customizations, complete the following tasks:

1. Download the customizations from the sandbox and verify them.
2. Publish the sandbox

#### Customize Enterprise Mobile Using Application Composer

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 customizations.
3. Navigate to Application Composer and select the Sales Application.
4. Under the Common Setup menu, select **Mobile Application Setup** and make your customizations. See the Customizing Oracle CX Enterprise Mobile: Worked Example topic for more information about customizing Enterprise Mobile.

#### Download Customizations From the Sandbox

1. After making your customizations, keep the Oracle Sales Cloud web application 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 Enterprise Mobile on your mobile device and sign in as a user that will be able to view your customizations. 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 customizations from the sandbox.
4. Select the sandbox that contains your customizations and tap **Save**.
5. Check your customizations. When you’re happy with your customizations you need to publish the sandbox.

### Publish the Sandbox

1. Sign in to the Oracle Sales Cloud web application as the same user you used to make the customizations.
2. Publish your sandbox to distribute your customizations to all Enterprise Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

---

### Oracle Sales Cloud Mobile

---

### Oracle Sales Cloud Mobile Extensibility: Explained

Application Composer lets implementors customize 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 customizations 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.

---

### Customizing Oracle Sales Cloud Mobile

This procedure shows you how to customize Oracle Sales Cloud Mobile for a specified sales object using Application Composer. For more information about Application Composer, see the Customizing Sales guide.

First you must open or create a sandbox, and then you customize 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 customizations. You might need to make a sandbox active, or create a sandbox, if a suitable sandbox does not exist.

#### Customizing Oracle Sales Cloud Mobile Using Application Composer

1. Open Application Composer by using the Navigator menu, and selecting **Application Composer** under the Tools category.
2. Select the application you want to customize within Application Composer.
3. Within the application you chose in step 1, select the custom or standard object you want to customize.
4. Select the **Pages** node in the navigation tree.
5. Select the **Mobile Pages** tab to see the mobile configuration options for the parent and its child objects.

6. 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.

7. Navigate through the configuration wizard to customize your selected object.

### Configuring Your Mobile Springboard

You can add or remove object icons from the Sales Cloud Mobile application as follows:

1. Within Application Composer, choose **Sales** from the Applications menu.
2. Open the **Mobile Application Setup** page, either by clicking the page link under the Common Setup list, 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 custom dynamic choice list fields in Application Composer. Both standard and custom dynamic choice list fields can be displayed on Oracle Sales Cloud pages, as well as on Oracle Sales Cloud Mobile pages.

In this topic, we’ll review 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, you’ll 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, you’ll 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 custom 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.
2. Create a dynamic choice list field based on that same object.
3. Add the field to your Mobile pages.

Let’s review each step in depth:

1. Configure the Mobile picker page, also known as a search and select 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. Select the application where your object exists, either Sales or Common.
c. Under the Objects navigation tree, expand the tree structure for your object.
d. Click the Pages node.
Click the Mobile Pages tab.

In the Picker region, click the Create Mobile Picker link for your object.

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.

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: "Customizing 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. Select the application where your object exists, either Sales or Common.
   c. Under the Objects navigation tree, expand the tree structure for your object.
   d. Click the Pages node.
   e. Click the Mobile Pages tab.
   f. 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.
g. 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 Customizations Using a Sandbox: Worked Example

Test all of your customizations in a sandbox before publishing them to the main application. Sandboxes are standalone environments where you define and test customizations before deploying the customizations to the production application.

Use Application Composer to customize Oracle Sales Cloud Mobile pages or objects in a sandbox environment, and then view your customizations 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 region, 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 customizations.

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 Tools category in the Navigator menu.
2. Select the application that you want to customize within Application Composer.
3. Select the parent object that you want to configure.
4. Select the Pages node in the navigation tree.
5. Select the Mobile Pages tab to see the mobile configuration options for the parent and its child objects.
6. 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.

7. Sign out of the Oracle Sales Cloud application.

Checking Your Customizations 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 customizations.
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 customized to ensure that they're working as expected.
5. Publish your sandbox to distribute your customizations to all Oracle Sales Cloud Mobile users. For more information about publishing sandboxes, see the Publishing Sandboxes chapter of the Oracle Sales Cloud - Customizing Sales guide.

Related Topics

- Sandboxes: Explained
- Using Sandboxes: Explained
- Sandboxes: How They Work with Some Customizations and Features
8 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 customize existing objects or create new custom ones. Read this chapter to learn about how and where you can use Groovy scripting in Application Composer.

🔗 Note: 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 customize 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 utilize 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’ll 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 customize 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 run time 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 the After Insert In Database and After Update In Database triggers.

  For example, consider a Contact object with a OpenTroubleTickets field that needs to be updated any time a trouble ticket is created or modified. You can create two triggers on the TroubleTicket object using the After Insert In Database and After Update In Database trigger events. When those events occur, your two triggers 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 invoke 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.
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 with the Common application, 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 specify one or more typed parameters that the caller will be required to pass in, when the function is invoked.
7. Specify the body of the function.
8. Validate and save your function.

Examples of Predefined Global Functions

This table lists the global functions that are provided with the Common application.

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>
</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 programmatically accesses that business object’s rows. For example, a common task that you’ll do with this new view object instance is to query some data. Do this by calling the `findByPrimaryKey()` 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 with the Common application.

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 system knows to run the script 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 delivered with the Common application, 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>
<tr>
<td>CodeAssignment</td>
<td>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.</td>
<td>Refer to the Trading Community Classification Code Assignment in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
<tr>
<td>Standard View Object</td>
<td>Description</td>
<td>Typical Attributes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>--------------------</td>
</tr>
<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>Access this Customer Contact Profile extensible object as a child of the Account, Contact, or Household objects.</td>
<td></td>
<td></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>The customer hierarchy and party hierarchy are modeled as trees.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FndTreeNode</td>
<td>Use this object to determine the parent/child relationships for a given hierarchy.</td>
<td>TreeStructureCode, TreeCode, TreeVersionID, TreeNodeID, ParentTreeNodeID, Depth, ChildCount, ParentPk1Value</td>
</tr>
<tr>
<td>The hierarchy for a given version is stored in this object.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FndTreeNodeRf</td>
<td>Use this object in scripting to easily access the flattened version of the given hierarchy version.</td>
<td>TreeStructureCode, TreeCode, TreeVersionID, TreeNodeID, Pk1Value, AncestorPk1Value, Distance, IsLeaf</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>Address is the intersection of location and party. Address fields like city, state, and country are stored in the location. These fields are made available in the Address object for read-only purposes. Use this object if you need write access to location fields in scripting.</td>
<td></td>
<td></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>
<tr>
<td>Standard View Object</td>
<td>Description</td>
<td>Typical Attributes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>PersonParty</td>
<td>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.</td>
<td>Refer to the Trading Community Person Details in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
<tr>
<td>PersonProfile</td>
<td>Access this Contact extensible object as a child of a PersonParty row or directly get the profile if you have a PartyID.</td>
<td>Refer to the Contact object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
<tr>
<td>Relationship</td>
<td>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.</td>
<td>Refer to the Relationship object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
<tr>
<td>Resource</td>
<td>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.</td>
<td>Refer to the Trading Community Resource Profile in the SOAP Web Services for Oracle Sales Cloud guide.</td>
</tr>
</tbody>
</table>

**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 table below. 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 supported classes and methods listed below 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><code>getLocale()</code></td>
<td><code>oracle.adf.share</code></td>
</tr>
<tr>
<td></td>
<td><code>getSecurityContext()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getUserRoles()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>isUserInRole()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getArray()</code></td>
<td><code>oracle.jbo.domain</code></td>
</tr>
<tr>
<td></td>
<td><code>getElemType()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getList()</code></td>
<td></td>
</tr>
<tr>
<td>ArrayList</td>
<td><code>getAttributeKind()</code></td>
<td><code>oracle.jbo</code></td>
</tr>
<tr>
<td></td>
<td><code>getIndex()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getJavaType()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getName()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getPrecision()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getProperty()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getScale()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getUIHelper()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getUpdateableFlag()</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>isMandatory()</code></td>
<td></td>
</tr>
<tr>
<td>Arrays</td>
<td><code>any constructor</code></td>
<td><code>java.util</code></td>
</tr>
<tr>
<td></td>
<td><code>any method</code></td>
<td></td>
</tr>
<tr>
<td>Arrays</td>
<td><code>any constructor</code></td>
<td><code>java.util</code></td>
</tr>
<tr>
<td></td>
<td><code>any method</code></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>getAttributeName()</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>
</tbody>
</table>
## Class Name | Allowed Methods | Package
--- | --- | ---
**Blob** | Any constructor<br>Any method | java.sql
**BlobDomain** | Any constructor<br>getBinaryOutputStream()<br>getBinaryStream()<br>getBufferSize() | oracle.jbo.domain
**Boolean** | Any constructor<br>Any method | java.lang
**Byte** | Any constructor<br>Any method | java.lang
**Calendar** | Any constructor<br>Any method | java.util
**Char** | Any constructor<br>bigDecimalValue()<br>bigIntegerValue()<br>booleanValue()<br>doubleValue()<br>floatValue()<br>getValue()<br>intValue()<br>longValue() | oracle.jbo.domain
**Clob** | Any constructor<br>Any method | java.sql
**ClobDomain** | Any constructor<br>toCharArray() | oracle.jbo.domain
**Collection** | Any constructor<br>Any method | java.util
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<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()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>clearAttributes()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raise()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raiseLater()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>warn()</td>
<td></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() getErrorCode()</td>
<td>oracle.jbo</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() 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>
</tbody>
</table>

"oracle.jbo"
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Allowed Methods</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<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>LinkedHashSet</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></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></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>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>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RandomAccess</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ref</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResourceBundle</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td>getAttribute()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>getAttributeHints()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getKey()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getLookupDescription()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getOriginalAttributeValue()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPrimaryRowState()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getSelectedListDisplayValue()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getSelectedListDisplayValues()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isAttributeChanged()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isAttributeUpdateable()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>revertRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>revertRowAndContains()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setAttribute()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>validate()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RowId</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RowIterator</td>
<td>createAndInitRow()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>createRow()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>findByKey()</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>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>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>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>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>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>
<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>sum()</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>getUserProfile()</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, Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>Any constructor, Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>StringBuilder</td>
<td>Any constructor, Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>StringTokenizer</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Struct</td>
<td>Any constructor, Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>Struct</td>
<td>getAttribute(), setAttribute()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td>StructureDef</td>
<td>findAttributeDef(), getAttributeIndexOff()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Time</td>
<td>Any constructor, Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Timer</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>TimerTask</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Time stamp</td>
<td>Any constructor, Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>Time stamp</td>
<td>Any constructor, compareTo()</td>
<td>oracle.jbo.domain</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>dateValue()</td>
<td>getValue()</td>
<td>java.util</td>
</tr>
<tr>
<td>stringValue()</td>
<td>timeValue()</td>
<td>java.util</td>
</tr>
<tr>
<td>timestampValue()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TreeMap</th>
<th>Any constructor</th>
<th>java.util</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TreeSet</th>
<th>Any constructor</th>
<th>java.util</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UUID</th>
<th>Any constructor</th>
<th>java.util</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UserProfile</th>
<th>getBusinessCity()</th>
<th>oracle, adf, share, security, identitymanagment</th>
</tr>
</thead>
<tbody>
<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></td>
<td>getEmployeeNumber()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getEmployeeType()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFirstName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getGUID()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getGivenName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getHomeAddress()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getHomePhone()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getInitials()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getJpegPhoto()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLastName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getMaidenName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getManager()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getMiddleName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getNameSuffix()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getOrganization()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getOrganizationalUnit()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPreferredLanguage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPrincipal()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperties()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getTimeZone()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getTitle()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUIAccessMode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUniqueName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUserID()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUserName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getWirelessAcctNumber()</td>
<td></td>
</tr>
<tr>
<td></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>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</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>ViewCriteriaRow</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></td>
<td>ensureCriteriaItem()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getConjunction()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isUpperColumns()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setConjunction()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setUpperColumns()</td>
<td></td>
</tr>
<tr>
<td>ViewObject</td>
<td>appendViewCriteria()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>avg()</td>
<td></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>createViewCriteria()</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>getMaxFetchSize()</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>max()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>min()</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>
</tbody>
</table>
Using Groovy Scripting: Examples

This topic contains examples of customizations that you can perform using Groovy scripts. Application Composer leverages Groovy to enable you to enhance your application customizations. 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. Select `Sales` from the `Application` list in the left pane.
3. Expand `Standard Objects` and then expand `Opportunity`.
   - Server Scripts Help Request page opens.
5. Under the Triggers tab, select `Action - Add` for Field Triggers.
   - Create Object Trigger page opens.
6. In the **Trigger** field, select **Before Update in Database**.
7. In the **Trigger Name** field, enter **TestPrintln**.
8. Under the **Trigger Definition** region, enter the following script in the expression text box:
   
   ```
   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.
9. Click the **Validate** icon. Confirmation appears when the script is parsed successfully.
10. Click **OK**.
11. In the **Navigator** menu, click **Opportunities**.
12. Click **Create Opportunity** in the left pane.
13. In the **Name** field, enter **Opportunity Trigger Test**.
14. In the **Win Probability (%)** field, enter **50**.
15. Click the **Save and Close** button.
16. On the Overview page, select the Opportunities tab.
17. Click the **Opportunity Trigger Test** link in the table.
   
18. In the **Win Probability (%)** field, enter **25**.
19. Click the **Save and Close** button.
20. Navigate to Application Composer.
21. Select **Sales** from the **Application** list in the left pane.
22. On the Overview page, click **Run Time Messages**.
23. Click the **Get Latest Log Messages** button.
24. 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. Select **Sales** from the **Application** list in the left pane.
3. Expand **Custom Objects**.
4. Expand **Help Request** and click **Fields**.
5. Under Custom tab, select **Action - Create**.
6. In the Select Field Type window, select **Text** and click **OK**.
7. On the Create Text Field page, enter **Justification** in the **Display Label**.
8. Click **Save and Close**.
9. On the Fields page, select **Action - Create** under the Custom tab.
10. In the Select Field Type window, select **Choice List (Fixed)** and click **OK**.
11. On Create Fixed Choice List page, enter **Priority** in the **Display Label**.
12. Under the List of Values region, click the **Create a New Lookup Type** icon.
13. In Create Lookup Type dialog, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Request Priority</td>
<td></td>
</tr>
<tr>
<td>Lookup Type</td>
<td>HR_PRIORITY</td>
</tr>
</tbody>
</table>

14. 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>

15. 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>

16. Click **Save**.
17. On Create Fixed Choice List page, select the **Fixed Value** option in the Default Value region, and then select **Important** as the default value.
18. Click the **Save and Close** button.
19. Under the Custom tab, click the **Justification** link.
20. Expression builder dialog opens.
21. Enter the following into the (script) text box:

   ```groovy
   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") }
   ```
22. Click the **OK** button.
23. Under Constraints region, select **Priority** from the **Depends On** list.
24. Click the **Save and Close** button.
25. Under Help Request in the left pane, click **Pages**.
26. Click the Edit Summary Form link. Edit Details Page Summary Form page opens.

27. 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.

28. Click the Save and Close button.
29. In the Navigator menu, click Help Request.
30. Click any active Help Request in the list to open its edit page.
31. Select Important from the Priority list.
32. Click the Save button.
The help request has been saved.
33. Now select Urgent in the Priority list.
An asterisk appears next to the Justification field that indicates a required field.
34. Click the Save button.
An error message appears, because you did not enter a value in the Justification field before saving.
35. Enter Laptop is on fire in the Justification field.
36. 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. Select Sales from the Application list in the left pane.
3. Expand Custom Objects.
4. Expand Help Request and click Fields.
5. Under Custom tab, select Action - Create.
6. In the Select Field Type window, select Text and click OK.
7. On the Create Text Field page, enter Executive Sponsor in the Display Label.
8. Click Save and Close.
9. On the Fields page, select Action - Create under the Custom tab.
10. In the Select Field Type window, select Check box and click OK.
11. On the Create Text Field page, enter Executive Sponsor Program in the Display Label.
12. Click Save and Close.
13. On the Fields page, click the Executive Sponsor link.
   Edit Check box Field: Executive Sponsor page opens.
14. Under Constraints region, select the **Updatable** check box and click the expression builder icon next to it. Expression builder dialog opens.

15. 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") } 
```

16. Click the **OK** button.

17. Under Constraints region, select **Executive Sponsor Program** from the **Depends On** list.

18. Click the **Save and Close** button.

19. Under **Help Request** in the left pane, click **Pages**.

20. Click the **Edit Summary Form** link.

21. Under Configure Default Summary region, move the **Executive Sponsor** and **Executive Sponsor Program** fields to the **Selected Fields** box.

22. Click the **Save and Close** button.

23. In the Navigator menu, click **Help Request**.

24. Click any active Help Request in the list to open its edit page.

25. Select the **Executive Sponsor Program** check box, and click the **Save** button.

26. Now clear the selection in the **Executive Sponsor Program** check box, and then click the **Save** button.

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. Select **Sales** from the **Application** list in the left pane.
3. Expand **Custom Objects**.
4. Expand **Help Request** and click **Fields**.
5. Under Custom tab, select **Action - Create**.
6. In the Select Field Type window, select **Text** and click **OK**.
7. On the Create Text Field page, enter **Corporate E-Mail** in the **Display Label**.
8. Click **Save and Close**.
9. In the left pane, click **Pages** under **Help Request**.

At this point, you can enter a name into the **Corporate E-Mail** field. You can’t enter anything in the **Executive Sponsor** field, because you have not selected the **Executive Sponsor Program** check box.
11. Under Configure Creation page region, move **Corporate E-Mail** to the **Selected Fields** box.

12. Click the **Save and Close** button.

13. Under Details Page region, click **Edit Summary Form**.

   Edit Details Page Summary Form page opens.

14. In the Configure Default Summary region, move the **Corporate E-Mail** to the **Selected Fields** box.

15. Click the **Save and Close** button.

16. In the left pane, click **Server Scripts** under **Help Request**.

17. Under Validation Rules tab, select **Action - Add** under the Field Rules region.

18. 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>Corporate E-Mail</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckValid</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid syntax in Corporate E-Mail. Enter a valid syntax for e-mail address</td>
<td></td>
</tr>
</tbody>
</table>

19. In the script textbox, enter:

   ```groovy
   newValue == null || newValue =~ /[\_A-Za-z0-9-]+([\_A-Za-z0-9-]+)*@[A-Za-z0-9]+([\_A-Za-z0-9-]+)*([\_A-Za-z]+)*/
   ```

20. Click the **Save and Close** button.

21. In the Navigator menu, click **Help Request**.

22. Click any active Help Request in the list to open its edit page.

23. Enter **mhoope.oracle.com** in the **Corporate E-Mail** field.

24. Click the **Save** button.

   The error message that you had specified appears, because the e-mail address does not have proper syntax.

25. Enter **mhoope@oracle.com** in the **Corporate E-Mail** field.

26. 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>GET</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td></td>
</tr>
<tr>
<td>PATCH</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>

For each method you want to expose, specify the information below.
<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name</td>
<td>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.</td>
</tr>
<tr>
<td>Format</td>
<td>Select a format for the method, based on what the selected web service returns.</td>
</tr>
<tr>
<td></td>
<td>- XML</td>
</tr>
<tr>
<td></td>
<td>- JSON</td>
</tr>
<tr>
<td></td>
<td>This information is provided by your web service provider or web service documentation.</td>
</tr>
<tr>
<td>Request Payload</td>
<td>Specify the object structure of the payload, if needed. You can do this in one of two ways:</td>
</tr>
<tr>
<td></td>
<td>- Directly provide the schema URL that represents the object structure.</td>
</tr>
<tr>
<td></td>
<td>- Provide a code sample in JSON or XML format. This is an optional parameter useful for displaying reference hints in the Expression Builder.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>This section is optional.</td>
</tr>
<tr>
<td>Response Payload</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>- Schema URL</td>
</tr>
<tr>
<td></td>
<td>A URL that provides a structure for the data but doesn’t include any values.</td>
</tr>
<tr>
<td></td>
<td>- Code Sample</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

**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 a response payload, enter `{ }` for JSON or `<a/>` for XML as the code schema sample.

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 Customizations

You can download the customizations you make in a "source" environment, and upload them into a "target" environment. This can save you time when working with customizations across multiple environments.

To do this, you’ll do the following:

1. Use the Customization Migration page to create a set of all customizations and extensions made to an application environment.
2. Then, download the customization set and upload it into another environment.

This is often referred to as the "customization set migration".
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 customization 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 customize 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>
</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>")
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
}
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 above, 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

```json
{
  "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",
```
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.)

```groovy
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 customizing 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:

   `?q=expression1;expression2`

   For example, maybe you want to retrieve all contacts in NY who belong to a specific department.

   `?q=Deptno>=10 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:

   `?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 above, 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 below 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>{ &quot;items&quot; : [ ], &quot;count&quot; : 0, &quot;hasMore&quot; : false, &quot;limit&quot; : 25, &quot;offset&quot; : 0, &quot;links&quot; : [ {</td>
</tr>
</tbody>
</table>
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.

```
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.

```
Method Name | POST
Format      | JSON
```
In the next two examples, you will create a POST request to create a new contact address using the REST endpoint that you registered above. 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: 'TestOrganization1']
    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
```
```groovy
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:
   ```groovy
   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>
</tbody>
</table>
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)
    println("Output:"+tickets)
}catch(Exception e){
    println("Headers:"+conn.responseHTTPHeaders)
    println("Status:"+conn.statusCode)
    println("Error:"+e)
}
```

**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:

```java
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:

```groovy
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:

<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:

    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:

    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):

    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:

    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:


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 Customizations
You can download the customizations you make in a "source" environment, and upload them into a "target" environment. This can save you time when working with customizations across multiple environments.
To do this, you will do the following:

1. Use the Customization Migration page to create a set of all customizations and extensions made to an application environment.
2. Then, download the customization set and upload it into another environment.

This is often referred to as the "customization set migration".

See: "Moving Customizations".

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 customization set to the target environment, you must re-create 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.

```groovy
// 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.

```groovy
// 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/NoSecuritywsdl">http://external-hosted7101/MathsWS-Model-context-root/NoSecuritywsdl</a></td>
</tr>
</tbody>
</table>

Note: The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider.
Decisions to Consider | In This Example
--- | ---
Which web service method will be called from the Groovy script? | getSum
This method returns the sum of two integer argument values.

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>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/">http://external-hosted7101/</a> MathsWS-Model-context-root/ Wss11UsernameWithMessageProtectionSecuritywsdl</td>
</tr>
<tr>
<td></td>
<td>This WSDL file specifies the desired message protection security 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></td>
<td>This method returns the sum of two integer argument values.</td>
</tr>
<tr>
<td>What will the server encryption alias name be?</td>
<td>serverenckey</td>
</tr>
<tr>
<td>Is it required to ignore the time stamp in the response from the web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td></td>
<td>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 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 mathws 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 invoke 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><strong>Note:</strong> 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>
<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></td>
<td>This method returns the sum of two integer argument values.</td>
</tr>
<tr>
<td>Is it required to ignore the time stamp in the response from the web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td></td>
<td>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. 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.

![Error Message]

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.

![Create SOAP Web Service Connection]

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 invoke 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 Fusion Applications 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/">https://internal-hosted7102/</a> MathsWS-Model-context-root/UsernameTokenOverSSLSecuritywsdl</td>
</tr>
<tr>
<td>This WSDL file specifies the desired SSL authentication scheme.</td>
<td></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>This method returns the sum of two integer argument values.</td>
<td></td>
</tr>
<tr>
<td>Is it required to ignore the time stamp in the response from the web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td>To ignore the time stamp, you select the Disable Time Stamp Verification check box.</td>
<td>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 `mathws` 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 Fusion Applications 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><code>mathws</code></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? | http://internal-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl
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 mathws 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 Fusion Applications 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 and add to the system.
Using a Script to Write Messages to the Log

To write messages to the diagnostic log, use the `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, select the web application from the Application list of values.
2. Go to the Common Tasks pane on the bottom left side, then click Run Time Messages.
3. 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:
   
   o Step Over
   
   Review one statement in the selected script at a time.
   
   o 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**.
   
   o 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.

### FAQs 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>);
```
9 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 a custom 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 topic.

- **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 topic.

- **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 topic.

- **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 topic.

- **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 topic.

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**.

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 customize 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 customizations you perform in Application Composer (such as creating new custom fields, objects, triggers, and so on) inside a sandbox are immediately taken into consideration when an object workflow is executed. The exception is that custom fields used as tokens in e-mail templates must be published before they can be used.

For more information on how you customize and extend Oracle Sales Cloud using Application Composer, refer Extending Oracle Sales Cloud: How It Works topic.
Expression Builder

An expression builder supports building logical and math operations, including field lookups that you can optionally use to define trigger condition. 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 using which you can define groovy conditions for object workflows.

Use the expression builder to write Groovy-based application logic that determines when an object workflow is triggered.

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:

```
Status=='IN_PROGRESS' && BudgetAvailableDate==Today() + 30
```
Example 2:

```java
if (isAttributeChanged('PrimaryContactPartyName') &&
    PrimaryContactPartyName == 'Business World') return true; else return false;
```

Example 3:

```java
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.


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'll 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 E-Mail Notification: How They Work Together

This topic covers how you create e-mail notifications action when creating object workflows. E-mail notifications send out automated alerts to the specified recipients when the associated object workflow is triggered. You can configure e-mail 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 e-mail alerts to the resources who are assigned to that opportunity.

Here's a summary of how you typically configure an e-mail 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: E-Mail Notification page, provide a **Name** for the e-mail notification action.
6. Search and select an existing e-mail template or create one.

When you create a template, you specify action-related text and field tokens that are populated at run time. See E-Mail 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 e-mail notifications and e-mail templates.

E-Mail Templates

Email notifications are based on e-mail templates, which define the layout of the e-mails and ensure that the notifications triggered by the same type of business event have consistent look and feel. See E-Mail Templates: Explained topic for more information on creating e-mail templates.

Recipient Types

The recipients available for sending the e-mail 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

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 custom text fields that store e-mails, you can select those fields to send e-mail notifications (for example, Primary Contact E-Mail under Opportunity object). The custom text field can store a single e-mail address or multiple comma-separated e-mail 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 e-mail 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 e-mail 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 e-mails.

- **Specific e-mail addresses**: Enter individual e-mails 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 system performs an automatic search after you type in your search terms and click or tab out of the search box.

**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.

- 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.
- 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

**Note:** File attachments cannot exceed 10 MB.
substitute them into the direct links as tokens. The following objects are supported: Opportunity, Lead, Account, Contact, Household, Activity, and any custom object. 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 customizing 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:

   o Select a Subject and Description, as appropriate, for your task notification.
   o 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.
   o Select the Owner and Resources for a 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. An owner can be manager of owner, record creator, and so on.

   For example, when an opportunity reaches a Close Date and the status is Open, you can use the task creation action to assign a follow up task to the owner of the opportunity.

   o 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.
   o Assign a Priority to the task. The default priority is Medium.

9. If a standard or a custom object 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 or Custom object 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’ll 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 a custom 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 a custom 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, navigate the Sales application.
2. Select the Activity object.
3. Select Fields.
4. Create a custom field of type Choice List (Dynamic).
5. Complete the general steps to configure a new dynamic choice list field. For example, set the display label for the field.
6. In the Related Object list, select the Asset object.
7. 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.

8. Click Submit.

Creating a Relationship

In this example, let’s create a relationship between the Activity and Asset objects.

1. In Application Composer, navigate to the Sales application.
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-messages 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/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:documentation/>
     <wsdl:operation name="processOutboundMessage">
       <wsdl:input message="tns:OutboundMessageService_processOutboundMessage"/>
       <wsdl:output message="tns:OutboundMessageService_processOutboundMessageResponse"/>
       <wsdl:Fault message="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:operation>
   </wsdl:binding>
</wsdl:definitions>
```
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<wsdl:fault name="ServiceException">
<soap:fault name="ServiceException" use="literal" encodingStyle=""/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="OutboundMessageService">
<wsdl:port name="OutboundMessageServiceSoapHttpPort" binding="tns:OutboundMessageServiceSoapHttp">
<soap:address location="http://adc2111013:7101/OMInterface/OutboundMessageService"/>
</wsdl:port>
</wsdl:service>
</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 $OBJECT_NAME$.

```xml
//Sample: OutboundMessageService.xsd

<schema elementFormDefault="qualified" targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
  xmlns:ns0="http://xmlns.oracle.com/adf/svc/errors/" xmlns:ns1="$OBJECT_TARGET_NAMESPACE$"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://xmlns.oracle.com/adf/svc/types/" schemaLocation="BC4JService.xsd"/>
  <import namespace="$OBJECT_TARGET_NAMESPACE$" schemaLocation="$OBJECT_NAME$.xsd"/>
  <import namespace="http://xmlns.oracle.com/adf/svc/errors/" schemaLocation="ServiceException.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.
Related Topics

- SOAP Web Services for Oracle Sales Cloud

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 a custom object 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.

### Object Workflows and Business Processes

**How They Work Together**

- **Business Process Composer (Approval Flows)**
  - Edit Approval Flow
  - Create Approval Flow
  - Seeded Template

- **Application Composer (Object Workflows)**
  - Define Object Workflow Trigger Condition
  - Configure Business Process Flow Event Action

- **BPM Worklist (Approval Requests)**
  - Lists Approval Request
  - Action (Accept/Reject)

---

- **Approval Process Triggered**

---

- **Deploy Project**
  - Call Deployed Project
  - Map Input Parameters

---

- **Oracle Sales Cloud**
- **Customizing Sales**
  - **Chapter 9**
  - **Creating Object Workflows**

---

- **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.

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**

- **Project Name:** ExtnBusinessProcessComposite

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>'hussanj'</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>'approvalStatus_c'</td>
</tr>
<tr>
<td>emailAddress</td>
<td>'<a href="mailto:ola@pinnacle.com">ola@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**.

This figure shows the Create Business Process Flow dialog.
• 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.
• Using the Search icon on the Create Action: Business Process Flow page.

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.

You can only edit the existing templates 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.

This figure shows the ExtnBusinessProcessComposite template.

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 sign in name of the person who should receive the approval request.</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 emails on the outcome of the approver action.</td>
</tr>
</tbody>
</table>
## Parameter | Description
--- | ---
title1 | Optionally append more information to the title of the notification.
title2 | Optionally append more information to the title of the notification.

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 customize some of the notifications, but with some limitations on what you can customize and how.

For more information on the types of notifications available and how you can customize 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.


**Supplying 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 ExtnBusinessProcessComposite 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 Customizing 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 Customizing 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>'jsmith'</td>
<td>Specifies the sign in name of the person who receives the approval request.</td>
<td>Literal</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:comlopez@pinnacle.com">comlopez@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 the 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 a custom 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.

\[\text{Note:} \] 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).

   The group can be static or dynamic.

5. Save the group.

### Creating the Approval Flow

Next, you’ll 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. At the bottom of the screen, 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>
<tr>
<td>Inputs</td>
<td>GroupApproval</td>
<td>Outputs</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>taskOwner</td>
<td>errorAssignee</td>
<td>outcome</td>
</tr>
<tr>
<td></td>
<td></td>
<td>taskOutcome</td>
</tr>
</tbody>
</table>

11. Click **Apply changes**.
12. Save the process.

### Deploying the Project

Finally, you'll 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 Username</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 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, select the application (for example, Sales).
3. Click Object Workflows.
4. Enter a name for the object workflow, and select an object (for example, Opportunity).
5. Click Create.
6. Click the Create icon next to Business Process Flow.
7. Enter a name for the flow.
8. Click Search next to Project Name.
9. Select GroupApproval from the list and click OK.
10. 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’ll 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’ll 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 Username</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, select the application (for example, Sales).
3. Click **Object Workflows**.
4. Enter a name for the object workflow, and select an object (for example, Opportunity).
5. Click **Create**.
6. Click the **Create** icon next to **Business Process Flow**.
Creating Object Workflows

7. Enter a name for the flow.
8. Click Search next to Project Name.
9. Select GroupApproval from the list and click OK.
10. 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. In supervisor hierarchy approval, WHAT?). 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’ll 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></td>
<td>title</td>
</tr>
<tr>
<td>taskOwner</td>
<td></td>
<td>taskOwner</td>
</tr>
<tr>
<td>taskErrorAssignee</td>
<td></td>
<td>errorAssignee</td>
</tr>
</tbody>
</table>
In the inputs to the SerialSupervisorHierarchyApproval human task, you can provide the starting approver, upto approver or number of levels.

11. Click **Apply changes**.
12. Save the process.

### Deploying the Project

Finally, you'll 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 Username</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, select the application (for example, Sales).
3. Click **Object Workflows**.
4. Enter a name for the object workflow, and select an object (for example, Opportunity).
5. Click **Create**.
6. Click the **Create** icon next to **Business Process Flow**.
7. Enter a name for the flow.
8. Click **Search** next to **Project Name**.
9. Select **SerialSupervisorHierarchyApproval** from the list and click **OK**.
10. 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. Select Sales from the Application list.
5. In the left pane, expand Standard Objects.
6. Expand Sales Lead.
7. Click Fields.
   Fields page opens.
8. Under the Custom tab, select Action - Create.
   Select Field Type dialog box appears.
9. Select Text.
10. Click OK.
    Create Text Field page opens.
11. In the Appearance region, enter Large Deal Approval in the Display Label field.
12. Click Save and Close.
13. In the left pane, select Pages under Sales Lead.
    Sales Lead: Pages page opens.
14. Under Desktop Pages tab, click Create Lead.
    Create Lead page opens.
15. In the Configure Detail Form region, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
16. 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.
17. Under Desktop Pages tab, click Show Details.
    Show Details page opens.
18. In the Configure Detail Form region, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
19. 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 Username</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).

   
   **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:** 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 Tools heading.
4. Select Sales from the Application list.
5. In the left pane, expand Standard Objects.
7. Click Fields.

The Fields page appears.
8. On the Custom tab, select Create from the Action menu.

The Select Field Type window appears.
9. Select Text.
10. Click OK.

The Create Text Field page appears.
11. In the Display Label field in the Appearance region, enter Large Deal Approval.

The applications 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.
12. 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.
3. Select Create from the Actions menu.
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:

```groovy
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.

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**.
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.

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'll 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’ll 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 customize and extend your Oracle Sales Cloud application. In the left pane, the default Application shown is Common, which hosts common objects such as Contact, Resource, Organization, and so on. In this activity, you are configuring a workflow for the Opportunity object; therefore, you’ll select Sales from the Application list.

3. In the left pane, click the Application list, then click the **Sales** list item.
4. 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’ll create a workflow.

5. Click the **Actions** menu below the Search region.
6. Click the **Create** menu item.

   The Create Object Workflow page appears.

7. You must first select an object for which you are creating a workflow. Click the **Object** list.
8. Click the **Opportunity** list item.
9. Enter “Budget Date Revised” in the **Name** field.
10. In the Event Point and Condition region, click the **When a record is updated** option.
11. You’ll 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.

12. Before you proceed, ensure that the **Fields** tab is selected.
13. 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.
14. Insert Date Budget Available into the expression builder. Click the **Insert** button.
15. 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'll create is updated.

The following script has been written for you using the BudgetDateAvailable field:

```java
if(isAttributeChanged('BudgetAvailableDate') && 'BudgetAvailableDate' != null && contains(Name, ' 50 Solar Green Servers')) { return true; } else { return false; }
```

16. Validate your script. Click the Validate button.
17. Click the OK button.
18. You have set the event point and trigger condition for your object workflow. You'll now create a Field Updates event action. In this event action, you'll 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'll 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'll 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'll 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'll 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'll need to adjust trailing or leading text-spaces as required.
8. Enter "Opportunity Budget Available Date is changed. Follow up with 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. Accept the default 0 (zero) in the **Days** field.
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'll now create the E-Mail Notification event action. Before you create this event action, you must first create an E-Mail Template that you'll 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'll create an e-mail template, which you'll 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'll 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'll 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.
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’ll 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’ll now create.

5. Creating an Opportunity Record

In this section, you’ll trigger the object workflow that you created in the previous steps by entering the budget availability date for an opportunity. You’ll 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’ll 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:
   ◦ Show More area in the Summary region.
   ◦ Revenue Items region.
   ◦ Activity Center region.
17. Locate the following fields and note their current values:
   ◦ Close Date (in the Summary region).
   ◦ Strategic Value (under Show Less area in the Summary region).
   ◦ Level of Risk (under Show Less area in the Summary region).
   ◦ 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 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, committing 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 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’ll 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'll 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.

### 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.
10 Creating Custom Subject Areas

Overview

Use report subject areas and custom subject areas to build reports in BI Composer. A subject area is a set of entities, attributes, and measures. Oracle Sales Cloud contains predefined report subject areas, but you can create your own. Read this chapter to learn how you create and configure custom subject areas and then use them to build reports using BI Composer.

This chapter covers:

- The concepts and terminology used in custom subject areas.
- Creating or editing, configuring, securing, and publishing custom subject areas and the considerations involved.
- Creating and viewing reports using published custom subject areas through examples covered in this chapter.

Custom Subject Areas: Explained

A custom subject area is a report subject area that you create for building reports using Oracle Business Intelligence (BI) Composer. Oracle Sales Cloud contains predefined report subject areas, but also enables you to create your own subject area for your reporting needs. The report subject area that you create is called a custom subject area.

A custom subject area is a set of entities, attributes, and measures that represent information about the areas of an organization’s business. When you use the BI Composer to build reports, the reports display according to the criteria that you specify as part of the report creation flow.

Before you begin creating your own (custom) subject areas, you might want to review these topics to get a better understanding of the terminology:

- Objects in Custom Subject Areas
- Fields or Attributes in Custom Subject Areas
- Measures in Custom Subject Areas
- Date Leveling in Custom Subject Areas
- Implicit Fact Column in Custom Subject Areas
- Creating and Editing Custom Subject Areas
- Securing Custom Subject Areas
- Publishing Custom Subject Areas

Concepts and Terminology in Custom Subject Areas

Before creating a custom subject area, it is important to understand the concepts and terminology used in custom subject areas. These concepts are discussed in their related topics in detail.

- **Objects**: Objects within Oracle Sales Cloud represent an entity; for example, an opportunity.
  - Standard object: An object delivered with Oracle Sales Cloud is called standard object; for example, Opportunity is a standard object.
Custom object: A new object that is added during the deployment of Oracle Sales Cloud is called a custom object.

- **Fields or attributes**: A field, also called an attribute, is where entity information is stored in Oracle Sales Cloud. An opportunity name is stored in a field under the opportunity entity or object.
- **Measures**: Measures are a set of functions that you can apply on date, numeric, or currency type fields of the selected primary, child, or related objects while defining a custom subject area.
- **Date Leveling**: Date leveling is a date hierarchy representation between dates and the associated measures. For example, total revenue by month.
- **Implicit Facts**: An implicit fact column defines the join path that should be used when running a report with only dimension attributes from a subject area that has measures from different facts or entities.

### Objects in Custom Subject Areas: Explained

A custom subject area is a set of entities (objects), attributes, and measures that you use to build reports. This topic covers how you use various types of objects in Application Composer to create custom subject areas.

Every custom subject area has a primary object, which is the focus of any reports 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 Oracle Sales Cloud are standard objects.

For more information on the various types of objects in Oracle Sales Cloud, see Defining Objects Explained topic.

#### Primary Objects

A primary object is any reportable top-level object. You create a custom subject area based on this primary object. Additionally, the primary object is the focus of the report that you create based on the custom subject area.

The list of available primary objects includes all reportable objects, which are either top-level custom objects, or standard objects that are configured as reportable by the owning Oracle Sales Cloud application. After you save your custom subject area, you can’t change its primary object; however, you can create another custom subject area using a different primary object.

**Note**: You cannot include Notes and Tasks in a custom subject area. They are not reportable.

Based on how you want to configure your custom subject area, you can add one child to an object.

#### Child Objects

A child object is an object that has a one-to-many relationship with a parent object and can be a parent object of another child object. Add a child object to a custom subject area if you want your report to include data from both the primary object and its children. If an object’s parent object is already a child object (of another parent object) then the object is a grandchild object. Custom subject areas support parent-child-grandchild-grand child objects.
This figure shows the parent-child-grandchild-grand grandchild hierarchy.

![Create Custom Subject Area: Select Child Objects](image)

You can add a single child object to the primary object as required, as long as there are child objects available. If there are no child objects for the chosen primary object, the list that enables selecting child objects does not appear.

The parent-child-grandchild-grand grandchild hierarchy supports adding only up to three levels of child objects, for example, parent-child1-child1.1-child1.1.1.

**Note:** Once you publish a custom subject area, you cannot add or remove child objects.

**Related Objects**

A related object is any object with a many-to-one relationship with its parent object. Custom subject areas support objects related to parent, child, grandchild or grand grandchild objects. You can add one or more related objects to a custom subject area. If a many-to-one relationship exists in the Relationships component of Application Composer and if the object is enabled for reporting, then it will be available in custom subject areas as a related object.

For example, when configuring a custom subject area, you can select the Opportunity object, which is a primary object, and then click the **Select Fields** button. Select **Related object**, then select a related object from the drop-down menu.
This figure shows how you add related objects.

You can also use the Select Fields dialog box to add or remove opportunity fields, or add or remove related object fields from the custom subject area. Select the fields you want to add or remove, then move them to the appropriate list.

After you publish a custom subject area, you cannot remove related objects; however, you can create another custom subject area and then use the applicable related objects.

For an example of how you configure a custom field and create a custom subject area using that field, see Creating a Custom Object and Associating it With a Custom Dynamic Choice List Field: Worked Example topic.

**Fields or Attributes in Custom Subject Areas: Explained**

This topic explains the various types of fields or attributes that you use for configuring your custom subject area.

Fields or attributes store entity information in Oracle Sales Cloud. Fields can be standard or custom. Standard fields are delivered with Oracle Sales Cloud, and custom fields are the ones that you create.

**Types of Reportable Fields**

The field types you can use to create reports are as follows:

- Text
- Number
- Date
- Percentage
- Date time
When you create a custom field, you can create reports for the following data types:

- Boolean

**Note:** If you are using Boolean data type for fields other than check boxes, those fields are displayed as either 0 or 1 on your custom reports.

- Number
- Currency
- Date
- String
- Percentage
- Phone
- Date time
- Long text

Creating Reports Using Extension Fields

For key objects in Oracle Sales Cloud, predefined extension dimensions exist which include custom or extended fields. These fields enable you to create reports on extensions that are made to standard objects.

**Note:** Extension dimension fields are not available for reporting until custom fields have been specified.

Here's a summary of the steps for creating a report using extension dimension fields:

1. Using Application Composer, create custom fields for standard objects, and ensure that the custom fields are exposed on the user interface.
2. Publish the sandbox.
3. In the navigator menu, select Reports and Analytics under Tools to navigate to Oracle Business Intelligence (BI) Composer.
4. Select a real time or Oracle Transactional Business Intelligence (OTBI) subject area that includes the predefined extension dimension.
5. Create a report.

When you specify the columns for your report, you can select the extension fields from the extension dimension folder, which appears as the <Object Name> Extension folder. For example, Opportunity Extension.

The extension fields available for reporting vary by object type.
Using Hierarchies in Reports with Custom Subject Areas

Some of the pre-delivered subject areas include hierarchical structures, such as customer hierarchy, that enable reporting and aggregating up the hierarchy. You can use these hierarchies in your reporting when you join a custom subject area with these pre-delivered subject areas that include hierarchies.

For example, you might create a custom object named Ticket, which includes the Customer field as a dynamic choice list. You then create a custom subject area for Ticket, and create a report using this custom subject area and one of the pre-delivered subject areas that include a customer hierarchy.

You can now report on the hierarchy using aggregations, such as Number of Tickets.

The following hierarchies are supported:

- Resource
- Territory
- Customer
- Partner

For example, you could create a custom object named Ticket, which includes the Account field as a dynamic choice list. Create a custom subject area for Ticket, then use this custom subject area and one of the pre-delivered subject areas that include a customer hierarchy:

- Create a report using a pre-seeded subject areas such as Sales - CRM Customer and Contacts Real Time that contains the account hierarchy.
- Add the custom subject area to the report to join the two subject areas.
- Add the account hierarchy from the pre-seeded subject area.
- Add the metric, such as Number of Tickets, from the custom subject area.
The report will calculate aggregations, such as Number of Tickets, using the hierarchy.

Reports and Analytics

Search Ticket Count by Account

Ticket Count by Account

<table>
<thead>
<tr>
<th>Customer</th>
<th>Number of Tickets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pinnacle International</td>
<td>6</td>
</tr>
<tr>
<td>Pinnacle Airlines</td>
<td>2</td>
</tr>
<tr>
<td>Pinnacle International</td>
<td>1</td>
</tr>
<tr>
<td>Pinnacle Tech Solutions</td>
<td>3</td>
</tr>
</tbody>
</table>

Refresh - Print - Export

Measures in Custom Subject Areas: Explained

Measures are a set of functions. Measure columns allow business users to see a summary of how their business processes are working so far. Examples include a SUM of the Revenue in Euros, or a COUNT of the number of Opportunities worth over $500,000. The designer defines the aggregation function (SUM, COUNT, and so on) for custom measures, so end users don’t need to do so themselves when they create an analysis. You can apply these functions on fields of type Date, Numeric, or Currency.

Measures available to a particular type of field may differ depending on the field type. After you define the measures for the required fields and publish the custom subject area, you can select these fields and the applied measures when creating your report in the Oracle Business Intelligence Composer. You can only specify aggregate formulas to apply to a measure when creating a custom subject area. It is not possible to edit a measure in an already published custom subject area.

Here are some measures you can apply to fields of type numeric, currency, or date.

- For numeric and currency fields, a measure can be:
  - **All**

    Note: All is not a measure, but an option in the UI that selects all of the measures.

  - Sum: Calculates the sum of the values.
  - Average: Calculates the mean value.
• Count: Calculates the number of rows that are not null.
• Count Distinct: Calculates the number of rows that are not null. Each distinct occurrence of a row is counted only once.

**Note:** Although Count Distinct is usually used in cases requiring a count on a foreign key (because a count of distinct rows is what’s wanted), it is not required. If your requirements allow multiple instances of the same foreign key value to be counted multiple times, you can use Count rather than Count Distinct.

• Maximum: Calculates the highest numeric value.
• Minimum: Calculates the lowest numeric value.
• First: Selects the first occurrence of the item.
• Last: Selects the last occurrence of the item.
• Median: Calculates the middle value.
• Standard Deviation: Calculates the standard deviation to show the level of variation from the average.
• Standard Deviation Population: Calculates the standard deviation using the formula for population variance and standard deviation.

• For date fields, a measure can be:
  • All
  • Maximum
  • Minimum

You can select measures based on your reporting needs. For example, you can use measures to view product sales per store, state, or country. Or, to view the number of support tickets opened or closed per day, week, or month, and so on.

**Date Leveling in Custom Subject Areas: Explained**

Date leveling is a hierarchy representation of dates and associated measures, which enables you to view the data over different periods of time.
This figure illustrates the date hierarchy linking a year with its quarters, months, weeks, and days.

![Date Hierarchy Diagram]

To use this hierarchy, create a report that shows the total Opportunity Sales amount for each year. Drill down from year to show sales per quarter, sales per month, sales per week and sales per day, and then drill back up to sales per year. The date hierarchy aggregates the measures by the required hierarchy or leveling period.

To configure date leveling when defining a custom subject area, use the **Configure Date Leveling** step of the train stop to either allow or disallow leveling. You may need to expand the field list in the **Date** field to select or clear the Date Leveling check box, as applicable.

For more information on where the Configure Date Leveling step appears in the train stop, see Creating and Editing Custom Subject Areas: Explained.

### Creating and Editing Custom Subject Areas: Explained

This topic covers how you create and edit custom subject areas, and how you can activate or inactivate a custom subject area while editing.

You cannot modify a predefined report subject area that is delivered with Oracle Sales Cloud; however, you can create custom subject areas to meet your reporting needs.

**Note:** Before you create a custom subject area, review all the included subject areas to see if the one you want is already available.

### Creating a Custom Subject Area

You create custom subject areas using train stops that appear at the top of the page. These train stops enable you to move back and forth during the configuration process. You can also save your configuration at a logical point and then continue to create later.

**Note:** You can only create custom subject areas when you are not in an active sandbox.

To create a custom subject area using train stops:

1. Navigate to Application Composer.
2. Select an application from the Application list.
3. Click Custom Subject Areas on the Overview page of Application Composer.
4. Select Actions - Create.

This figure shows the first page in the guided process for creating custom subject areas.

Here are the steps in the train stops that you can use for configuring your custom subject area:

1. Define Custom Subject Area

   In this step, you provide the name for your subject area and select the primary object that is the basis for the reports you create later using the custom subject area. Subject areas usually have names or labels that correspond to the type of information they contain, such as service requests and orders. Display labels have the Custom: prefix added automatically.

2. Select Child Objects

   In this step, you select the child objects whose data you want to use in your reports. You can add child objects only if the primary object has child objects. Otherwise, the add icon is disabled. When you select child objects for your custom subject area, consider the following points:

   - For a one-to-many primary-child relationship, you can add a single child object if required. The parent-child-grandchild-great grandchild hierarchy supports adding only up to three levels of child objects, for example, parent-child1-child1.1-child1.1.1.
   - For a many-to-one primary-related relationship, you can add as many related objects as you want.

3. Configure Fields

   In this step, you select the fields that you want to display on your reports. You typically add at least one field from each of the objects that you have selected for your custom subject area.

   Select the desired measures to generate for number, date, or currency fields from all the available objects so that the subject area includes only those measures that you want to analyze. Also, define at least one measure.

   In the Measure Aggregations column, select an option from the list of predefined formulas that you can apply to the Measure field. When you select the formula, the application applies the selected formulas to the selected field and measures.
You can change the display labels of the fields that you select in this step. Additionally, you can use the Select Fields dialog to remove fields that belong to the primary object, or add fields from the related objects. The Select Fields dialog appears when you click Select Fields when configuring fields for your custom subject area.

This figure shows the Select Fields dialog.

After you publish your custom subject area, the fields you have selected for your subject area are automatically added to their owning object’s folder. If you have also defined measures, those fields are automatically added to the Facts folder.

For more information on measures, see Measures in Custom Subject Areas: Explained.

This figure shows the Facts folder and the object folder whose fields are selected.
4. Configure Marketing Segmentation

For more information on marketing segmentation, see Marketing Segmentation: Functional Overview and its related topics.

5. Configure Date Leveling

If required, select the Date columns for date leveling. For more information on date levelling, see Date Levelling in Custom Subject Areas: Explained.

6. Configure Security

Select the required security level for the Everyone Role Name, which is added by default, or add additional Role Names by clicking in the + icon and define the security level for each one of them.

The security definition here only controls who can access the custom subject area definition to create reports. It doesn’t control data visibility which is automatically controlled based on the user running the reports.

For more information on securing custom subject areas, see Securing Custom Subject Areas: How It Works topic.

7. Review and Submit
Review the custom subject area configuration for all added objects, attributes, and measures, and if satisfied, click **Submit**. If changes are required, click **Back** to navigate back to the required screen.

After you submit, the custom subject area configuration is prepared for publishing. You can create and submit a custom subject area either immediately or save and close the custom subject area at any point and submit it later. You must first submit a custom subject area for publishing before you can select it from within Oracle BI Composer. After you save or submit a custom subject area, you cannot modify its primary object.

To access the published custom subject area in BI:

- From the Navigator menu, select **Tools - Reports and Analytics**.
- In the Contents pane, click **Create**.
- Select the published custom subject area and start creating your report.
This figure shows a sample custom subject area. Note the custom: prefix.

**Editing Custom Subject Areas**

You can edit a published or saved custom subject area and then republish it when your changes are done. Modifying a custom subject area does not affect the reports that you had created using that custom subject area before making the changes. You can use the modified custom subject area if you need to enhance existing reports.

To edit a custom subject area:

1. On the Overview page of the Application Composer, click **Custom Subject Areas**.
2. From the **Application** list, select the application depending on the object you used for creating the custom subject area.
3. Locate the custom subject area that you want to edit, and click the Edit icon.

   You can filter out inactive custom subject areas in Application Composer by viewing custom subject areas in Active status. This is safer than deleting them, because the inactive subject areas are still available and can be found by searching.

4. Make the desired changes and then click **Submit** to republish the custom subject area.

While you can edit a custom subject area in any status, there are considerations on what you can or can’t do when editing. When editing a published custom subject area, it is not possible to:

- Change the primary object.
- Add or remove child objects.
• Remove previously added measures.
• Add more aggregation types for measures that are already published.

Note: You cannot modify a predefined report subject area that is delivered with Oracle Sales Cloud. Instead, you must create separate custom subject areas to meet your reporting needs. Before you create a custom subject area, be sure to review all the included subject areas to see if the one you want is already available.

Activating or Inactivating Custom Subject Areas

When editing custom subject areas, you can activate or inactivate custom subject areas when your reporting or business requirements change. This enables you to control what information is displayed on the reports that use the information from custom subject areas.

You can inactivate only those custom subject areas that are published and a status of OK, and can activate only previously inactivated custom subject areas.

To inactivate a custom subject area, select it in the list and then click the Inactivate button. To activate an inactive custom subject area, select it and click Activate. Note that if no custom subject area is selected in the list, the button doesn’t appear.

This graphic shows an active custom subject area selected, and the Inactivate button.

When searching for custom subject areas, you can filter out inactive custom subject areas in Application Composer by viewing only those in Active status. Inactivating a custom subject area is safer than deleting it, because the inactive subject areas are still available and can be found by searching.

Configuring Custom Subject Areas for Segmentation: Points to Consider

This topic outlines the setup required to target some organization contacts using criteria from a custom object that has a relationship with an Organization Customer (B2B). It also specifies some setup variations for configuring custom objects related to B2C Customers (Consumers) and custom objects related to contacts.
Configuring Custom Subject Areas in Segments

The following example is used to illustrate your task. You are a B2B company and want to target organization contacts in a marketing campaign. The contacts work for organizations that have placed an order with your company in the past month.

Before You Begin

The following lists the tasks that you must perform before you begin to configure the custom subject area for segmentation:

- Create a custom object named Orders and include custom fields such as date, amount, product and so on using the dynamic choice list. Select Account as the target object for the dynamic choice list to allow you to select the customer for which each new order is required. This action creates a relationship between Order and Account.
- Create a work area for the custom Order object and expose the necessary fields in the Overview, Create and Detail pages.
- Create data records for the Order object either through the UI, through Import or through web services. When creating records, make sure that you select the Account for which each Order is required.
- Create a custom subject area for the Order object and expose that subject area for segmentation. When selecting fields to include in the custom subject area you must choose the Party ID from the Account object.

Custom Subject Area Segment Configuration

Do the following to configure the custom subject area for segmentation:

1. Associate the subject area to one or multiple Target Levels.

   In this example, the custom object has a relationship to Account only, so it must be related to the Real Time Customers target level only. The Qualifying Identifier must be set to the same level as the target level (Customers Real Time) and the Mapped Field must be set to the Party ID of the Account object. This association ensures that Marketing Segmentation uses the Party ID to identify and count individual customers.


   Set the Qualifying Identifier to the same level as the target level (Real Time Customers) and set the mapped field to the Party ID of the Account object. This association ensures that Marketing Segmentation uses the Party ID as the parameter when querying the database for the set of customer data to include in the segment.

Using Custom Subject Areas in Segments

Now that the custom subject area has been set up and Marketing Segmentation has been configured, you can now use the custom subject area when creating segments. Using this example, the target for the segment is individual contacts that work for organizations which have an order. You need to create a segment that targets Real Time Customers and then nest (relate) the Real Time Contacts subject area.

Perform the following steps:

1. Navigate to the Audience work area and select Create Segment. Enter a name for your segment, select the Real Time Contacts target level, and click Save and Design. The Segmentation UI will be displayed.
2. Click the Select Another Target Level icon in the upper right corner (next to the Save icon) and select the Real Time Customers target level.
3. Select the Add/Remove Subject Areas icon in the upper left corner of the screen (in the Subject Area section next to Refresh icon). Select the new custom Orders subject area that you previously created.
4. Create the segment criteria for the set of Organizations that you want to target (based on orders, or customer attributes), then save the segment.
Note: This segment will not appear in Marketing Segmentation UI at any time. It is a nested segment used by the Contacts Real Time segment that is being created.

5. Once the Customer level segment is saved, click Go back to segment name link in the upper left of the Segment Designer section. The segment_name is the name of the segment that you are designing. This action returns to the original segment that is at the Contact level and the Customer level segment will be nested as the first criteria.

6. Add any other criteria necessary to the segment, at the customer or contact level (customer name, contact e-mail, and so on).

7. Save the segment and update counts.

The counts should reflect the number of contacts that are related to the companies which meet the Orders criteria (plus any other criteria entered into the segment).

Custom Objects Related to B2C Customers (Consumers)
The setup and process for creating the custom subject area for custom objects related to Consumers is almost identical to the process outlined in the previous example with a few exceptions as follows:

• When creating the custom object and inserting the dynamic choice list, the target object must be the Contact object. This object relates the Person Customer to the custom object (instead of an Account).

• When setting up the custom subject area, the Party ID of the Contact object must be added to the subject area in the Field selection step.

• When configuring segmentation on the custom subject area, the target level must be set to Real Time Consumers. The Qualifying Identifier must be set to Consumers Real Time and the Party ID from the Contact object must be used as the mapped field in both steps.

• A nested segment is not needed when you create a segment. The custom subject area is related directly to the person customer (consumer) so that object can be used in a segment that is targeting consumers.

Custom Objects Related to Contacts

The setup and process for creating the custom subject area for custom objects related to Contacts is almost identical to the process outlined in the previous example with a few exceptions as follows:

• When creating the custom object and inserting the dynamic choice list, the target object must be the Customer Contact Profile object. This object relates the Contact to the custom object (instead of an Account or Person Customer).

• When setting up the custom subject area, the Party ID of the Customer Contact Profile object must be added to the subject area in the Field selection step.

• When configuring segmentation on the custom subject area, the target level must be set to Real Time Contacts. The Qualifying Identifier must be set to Contacts Real Time and the Party ID from the Customer Contact Profile object must be used as the mapped field in both steps.

• A nested segment is not needed when you create a segment. The custom subject area is related directly to the customer contact (contact) so that object can be used in a segment that is targeting contacts.
Securing Custom Subject Areas: How It Works

You can secure a custom subject area by granting or revoking access rights from the role names that access the custom subject area. This topic covers how you can add or delete role names, or grant or revoke access rights from those role names.

You can also add role names from a predefined list and assign or revoke permissions.

Managing Role Names and Access Rights

While defining a custom subject area using the train stops, you can use the Actions list in the Configure Security step to manage role names and access rights as follows:

- Select and add role names for a custom subject area from a predefined list of role names. This predefined list also provides the description for each role name. You can also select and add multiple role names from this predefined list using either the Shift or Ctrl keys. Once you add a new role name, you can select appropriate access for that role name.

- Select and delete role names listed for a custom subject area. You can also select and delete multiple role names using either the Shift or Ctrl keys.

Note: You cannot delete the role name listed as Everyone.

- Read access is granted by default to each role name you add. If you want to revoke Read access from a listed role name, select No access for that role name.

Note: You can create custom subject areas even for the objects in which you do not have access to the data, which allows you to build custom subject areas without compromising data security.

Publishing Custom Subject Areas: Explained

This topic covers what happens when you submit a custom subject area for publishing, and what the submission statuses indicate.

After you successfully publish your custom subject area, you can start building reports using Oracle Business Intelligence (BI) Composer based on your published subject area.

What Happens When you Submit for Publishing

When you submit a custom subject area for publishing, two processes occur in the background. The first process is synchronous and creates Oracle Applications Development Framework (Oracle ADF) artifacts. You must wait until this first process is over. The second process is asynchronous and creates centralized metadata repository (RPD) fragments and submits them to the Oracle BI server.
A custom subject area can have one of the following statuses:

- **Pending**: This status indicates either of the following:
  - You saved and closed the configuration process for a custom subject area before submitting it for publishing.
  - A failure occurred in the background processes when creating Oracle ADF and RPD artifacts.
- **In Process**: This status indicates that the data is in the process of being published to Oracle BI.

**Note**: If the in-process status does not change to **OK**, even after multiple refresh attempts, then there could be an error in publishing. If an error occurs, the details are displayed, as well as information about how to fix problems, where applicable. These error status details allow you to pinpoint and fix problems quickly.

- **OK**: This status indicates that the custom subject area has been published successfully. You can use Oracle BI Composer to create reports using the objects, attributes, and measures that you have configured in the subject area.

---

**Associating Custom and Standard Objects and Creating Reports: Worked Example**

This example illustrates how to create a custom object and associate it with a standard object using a custom Choice List (Dynamic) field, and then create reports using both objects.

In this example, for each opportunity record, you want to include the option to associate the name of the partner who won that opportunity. Doing this lets you create a report showing a summary of selected opportunities (for example, all won opportunities in the previous quarter) along with the name of the winning partner for each opportunity.

This example covers:

1. Creating a custom object called Winning Partner.
2. Associating the Winning Partner object with the Opportunity object using a custom Choice List (Dynamic) field.
3. Creating a custom subject area using data from both the custom Winning Partner object and the standard Opportunity object.
4. Creating a report using the custom subject area.

---

**Creating a Custom Object**

In this step, you create a custom object called Winning Partner.

**Note**: Before making any customization in Oracle Sales Cloud, you must have a sandbox session active. For more information on sandboxes and how to use them, see Sandboxes: Explained.
To create a custom object:

1. Navigate to Application Composer.
2. On the Overview page, select Sales from the Application list in the left pane.
3. Click the create icon for custom objects.
4. Configure the custom object as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Enter or Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Label</td>
<td>Winning Partner</td>
</tr>
<tr>
<td>Plural Label</td>
<td>Winning Partners</td>
</tr>
<tr>
<td>Record Name Label</td>
<td>Partner Name</td>
</tr>
<tr>
<td>Record Name Data Type</td>
<td>Text</td>
</tr>
<tr>
<td>Object Name</td>
<td>WinningPartner</td>
</tr>
</tbody>
</table>

5. Click OK.
6. In the left pane, expand the Winning Partner node, and click Fields.
7. On the Fields page, select the Standard tab to view the standard fields that are automatically created for the new custom object. You can optionally add custom fields on the Custom tab.
8. Next, click Pages under the Winning Partner node.
10. Under Landing Page Layouts, copy the standard layout and edit it.
11. Click the Edit icon next to Summary Table.
12. Notice that under Configure Summary Table, Partner Name is specified as the Drill Down Column, and is included automatically in the Selected Fields list.
13. Click Cancel, then click Done.
15. Click the Edit icon next to Winning Partner Create.
16. Under Configure Details Form, notice that Partner Name is included in the Selected Fields list by default.
17. Click Cancel, then click Done.

Associating the Winning Partner Object with the Opportunity Object

In this section, you create a custom Choice List (Dynamic) field for the Opportunity object using the Winning Partner object.

To create a custom dynamic choice list:

1. Navigate to Application Composer.
2. Expand the Standard Objects tree in the left pane.
3. Expand the Opportunity node, and select Fields.
4. Under the Custom tab, select Actions - Create.
5. In the Select Field Type dialog, select Choice List (Dynamic) and click OK.
6. In the Create Dynamic Choice List: Basic Information page, specify the following:
## Creating Custom Subject Areas

In this step, you create a custom subject area to enable reporting on Opportunity and Winning Partner objects.

To create a custom subject area:

1. If you are working in a sandbox, exit the sandbox.
2. Navigate to Application Composer.
3. Select **Sales** in the **Application** list in the left pane.
4. Click **Custom Subject Area** in the Overview region.
5. Select **Actions - Create**.
6. In the Define Subject Area step, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Enter or Select</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Label</strong></td>
<td>Opportunity Winning Partners</td>
</tr>
<tr>
<td><strong>Primary Object</strong></td>
<td>Opportunity</td>
</tr>
</tbody>
</table>

7. Click **Next**.
8. Click **Next**.
9. In Configure Fields step, click **Select Fields**.

---

### Fields

<table>
<thead>
<tr>
<th>Display Label</th>
<th>Winning Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Winning</td>
</tr>
<tr>
<td><strong>Updatable</strong></td>
<td>Select</td>
</tr>
<tr>
<td><strong>Include in Service Payload</strong></td>
<td>Select</td>
</tr>
</tbody>
</table>

7. Click **Next**.
8. Under List Data Source, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Related Object</strong></td>
<td>Winning Partner</td>
</tr>
<tr>
<td><strong>List Selection Display Value</strong></td>
<td>Partner Name</td>
</tr>
</tbody>
</table>

9. Click **Submit**.

You have now defined a relationship between the Winning Partner custom object and Opportunity standard object.

10. Test your customizations to ensure they work properly, and then publish your sandbox.
In the Select Fields dialog:

a. Select the Related object option.
b. Select the WinningPartner_c custom object.
c. Move the Record ID and Partner Name fields to the Selected Fields box.
d. Click OK.

10. Select the desired measures.
11. Click Next.
12. Select the Date Leveling check box for the Date fields.
13. Accept the default Role Name security settings.
14. Click Next.
15. Review the custom subject area configuration.
16. Click Submit.

Wait until you see the page that shows the status of the custom subject area in a table. You may have to refresh the page until the status changes to OK.

Configuring a Report Using the Custom Subject Area

In this section, you create a report using the published custom subject area and then view the report.

You create a report using a guided process, wherein you can specify the fields and measure that you want to display.

To create a report:

1. In the Navigator menu, click Reports and Analytics under Tools category.
2. Click Create.
3. Under Select Subject Area, locate and select the published Custom: Opportunity Winning Partners custom subject area.
4. Use the guided process to configure your report as follows:
   a. In the Select Columns step, open each folder and select the data you want on your report.
   b. Click Next.
   c. In the Select Views step, enter a Title for your report, and specify the view options for Table, Graph, and Layout.
   d. Click Next.
   e. In the Edit Table step, specify table layout options.
   f. Click Next.
   g. Define the Table layout options.
   h. Click Next.
   i. In the Edit Graph step, specify how you want the graph to be displayed.
      You can select Preview check box to view how the graph looks like based on your current specifications.
   j. Click Next.
   k. In the Sort and Filter step, define how you want to sort and filter data that appears on the report.
   l. Click Next.
   m. In the Highlight step, if required, define the conditional formatting for the table.
   n. Click Next.
   o. Enter the Analysis Name as Opportunity Winning Partners. This is the name by which the report is saved.
5. Navigate to the folder where you saved the Opportunity Winning Partners report.
6. Click **Opportunity Winning Partners**, and then click **View** to see the report.

### Extensibility and Reporting: Example

In this example, you first add an attribute to a standard object and see how that attribute becomes automatically reportable. You then create a custom subject area using this attribute and create a report.

### Adding an Attribute to a Standard Object

In this example, you'll add a custom, fixed-choice field called **Strategic Value** to the opportunity object.

1. Navigate to Application Composer.
2. In the left pane, click the **Application** list, then click the **Sales** list item.
3. In the Objects region on the left, locate the opportunity object.
4. Click the **Expand** button of the Standard Objects tree.
5. Under the Standard Objects tree, click the **Expand** button of the Opportunity tree item.
6. Click the **Fields** link under the Opportunity tree item.

You are on the Fields page.

7. In the Custom tab, click the **Create** button.
8. In the Select Field Type dialog, click the **Choice List (Fixed)** option, then click **OK**.

You are on the Create Fixed Choice List page.

9. In the Appearance region, enter “Strategic Value” in the **Display Label** field.
10. Enter “Select whether the deal is strategic” in the **Help Text** field.
11. In the List of Values region, click the **Search and Select Lookup Type** button.
12. In the Search and Select: Lookup Type dialog, search for existing **Yes/No** fields.

In the Search region, enter “Yes” in the **Meaning** field.

13. Click **Search**.
14. In the search results, click the **Yes / No Lookup Type** cell.
15. Click **OK**.

You are back to the Create Fixed Choice List page. You will now set a default value for a new deal.

16. In the Default Value region, click the **Fixed Value** list.
17. Click the **No** list item.
18. In the upper right region of the page, click the **Save and Close** button.

You have added an attribute to a standard object.
Viewing Added Attribute as Reportable at Run Time

In this example, you’ll view the attribute that you added to the opportunity object, which is now reportable in the related subject area at run time.

1. Click the **Navigator** menu.
2. Click the **Reports and Analytics** link.

   ✍️ **Note:** You might need to click the more >> link first.

   You are on the Reports and Analytics page. Use the toolbar in the left pane to navigate to Oracle Business Intelligence Answers.

3. Click the **Browse Catalog** button.

   You are on the Oracle Business Intelligence Answers page. Use the Folders pane on the left to navigate to the extensions created for the opportunity object.

4. Double-click the Shared Folder tree.
5. In the Shared Folders tree, double-click the **Sales** tree item.
6. In the Sales tree item, double-click the **Subject Area Contents** tree item.
7. In the Subject Area Contents tree item, select **Sales - CRM Pipeline**.
8. In the main area of the page, click the **Edit** link under Pipeline.

   The left pane lists the standard and extended objects used for the Sales - CRM Pipeline subject area.

9. Locate the Opportunity Extension tree item to see the attribute you added.
10. Double-click the Opportunity Extension tree item.

   You’ve verified that a custom attribute is reportable at run time.

Creating a Custom Subject Area

In this example, you’ll create a custom subject area using the opportunity object.

1. If you are working in a sandbox, exit the sandbox.
2. Navigate to Application Composer.
   You’ll be creating a custom subject area using the opportunity object. Opportunity belongs to the Sales application, so you’ll select the Sales application for your activity.
3. In the upper left region of the page, click the **Application** list.
4. Click the **Sales** list item.
5. In the Overview region, click the **Custom Subject Areas** link.

   You are on the Custom Subject Areas page. You can use this page to search or create custom subject areas.

6. In the Search Results region, click the **Create** button.
7. Enter a name for the custom subject area that you are creating. Enter "Opportunity_Contact" in the **Label** field.
8. In the Primary Object region, click the **Primary Object** list.
9. Click the **Opportunity** list item.
10. In the upper-right region of the page, click the **Next** button.
11. In the upper-right region of the page, click the Add Child Object button.
12. In the Add Child Object dialog box, click the Child Object list.
13. Click the OpportunityContact list item.
14. Click the OK button.
15. In the upper-right region of the page, click the Next button.

You’ll now define measures for date and numeric fields for the Opportunity primary object. You don’t need measures applied to all Date and Numeric fields, which is currently selected by default, so you’ll first remove the default selection, then add measures to the fields you require for your custom subject area.

16. Ensure that the selected value in the Fields From list is Opportunity. Click the Actions menu.
17. Click the Deselect All Dates as Measures menu item.
   This action will deselect all Measures selected by default for the Date fields.
18. Click the Actions menu.
19. Click the Deselect All Numerics as Measures menu item.
   This action will deselect all Measures selected by default for Numeric fields.
20. You’ll now specify the fields you want to apply measures to. Click the Measure option for the Revenue field.

   In this activity, you won’t be applying measures for fields in the OpportunityContact child object.
21. In the upper-right region of the page, click the Next button.
22. You’ll now select fields to apply date leveling. In the Date Field Leveling table, click the Expand button of the Opportunity object.
23. Select the Allow Leveling option for the RevenueEffectiveDate field.
24. In the upper-right region of the page, click the Next button.
25. Leave the default role access of Read for Everyone. In the upper-right region of the page, click the Next button.
26. Review your custom subject area. In the upper-right region of the page, click the Save button.
27. In the upper-right region of the page, click the Submit button.
   A confirmation message appears.
28. Click the OK button.
   You’ve successfully created a custom subject area.

Creating a Report Using a Subject Area

In this example, you’ll create a report in Oracle Business Intelligence Composer (BI Composer) using the Sales - CRM Pipeline subject area.

1. Click the Navigator menu at the top.
2. Click the Reports and Analytics link under Tools.

   Note: You might need to click the more >> link first.

   The Reports and Analytics page appears.
3. In the left pane, click the Create button.
   From the Select Subject Area dialog box that appears, you must first select a subject area to build your report. In this activity, you are building a report using the Sales - CRM Pipeline subject area.
4. In the Select Subject Area dialog, locate and click the Sales - CRM Pipeline link.
Oracle Sales Cloud
Customizing Sales

Creating Custom Subject Areas

**Tip:** You can add more than one subject area to a report by clicking the Add/Remove Subject Areas icon in the top right corner of the subject area list. You can include both standard and custom subject areas in the same report.

5. You are in the Select Columns step of Oracle Business Intelligence Composer (or BI Composer) wizard. In the left box, click the **Expand** button of the Employee tree.

6. Add the following fields to the **Selected Columns** box on the right:
   - a. Employee Name under Employee tree.
   - b. Opportunity Name under Opportunity tree.
   - c. Sales Stage Name under Opportunity tree.
   - d. Status under Opportunity tree.
   - e. # of Opportunities from Pipeline Facts tree.

7. In the upper-right region of the page, click the **Next** button.

8. Enter "Opportunity Count By Sales Stage" in the **Title** field.

9. Click the **Graph** list.

10. Click the **Bar (recommended)** list item.

11. Click the **Preview** option on the right of the **Title** field.

12. In the upper-right region of the page, click the **Next** button.

You are in the Select Views step of the wizard.

13. From the **Group By** box, exclude the fields you do not want in your report. Notice how the **Preview** changes.

14. In the **Group By** box under the Graph Layout area, click the **Opportunity Name** option.

15. Click the **Move To** list adjacent to the Group By heading.

16. Click the **Excluded** list item.

17. In the Group by box, click the **Employee Name** option.

18. Click the **Move To** list adjacent to the Group By heading.

19. Click the **Excluded** list item.

20. In the upper right region of the page, click the **Next** button.

21. You are in the Sort and Filter step of the wizard. In this step, you will select the following filters for your report:
   - o Sales Stages to display in your report
   - o The Customer for which these Sales Stages should be displayed.

22. In the Filter region, click the **Add Filter** list.

23. Click the **Sales Stage Name** list item.

24. Click the **Operator** list of the Sales Stage Name filter.

25. Click the **is in** list item.

26. Click the **Value** list of the Sales Stage Name filter.

Select these options:
   - o 02 - Negotiation
   - o 07 - Closed
   - o Short List
   - o Solution Presentation

27. In the Filter region, click the Add Filter list to add another filter.

28. Click the **More Columns...** list item.

You are on the Select Column dialog box.
29. In the dialog, click the **Expand** button of the Customer tree.
30. In the Customer tree, click the **Customer Name** tree item.
31. Click the **OK** button.
   You are back to the Sort and Filter step.
32. In the Filter region, click the **Operator** list of the **Customer Name** filter.
33. Click the **is in** list item.
   You will now search for a customer name.
34. Click the **Search** button adjacent to the **Value** list of the **Customer Name** filter.
   You are on the Select Values dialog.
35. Ensure that the **Name** field contains begins with value. Enter "Pinnacle" in the **Search Criteria** field.
36. Click the **Search** button.
37. Click the **Pinnacle Technologies** item in the **Available** box.
38. Click the **Move selected items to other list** button in the middle.
39. Click the **OK** button.
40. In the upper right region of the page, click the **Next** button.
   You are in the Save step of the wizard.
41. Enter "Opportunity Count By Sales Stage" in the **Report Name** field.
42. In this activity, you will save your report in only **My Folders**. In the **Save In** area, click the **My Folders** tree.
43. In the upper right region of the page, click the **Submit** button.
   A confirmation message appears.
44. Click the **OK** button.
45. You can now view the report you just created. Click the **Expand** button of the **My Folders** tree.
46. Locate the report you just created.
   You've successfully created a report in BI Composer using CRM - Sales Pipeline subject area.

**Custom Subject Areas: Frequently Asked Questions**

**Can I change a custom subject area's primary object?**

No. Once you save a custom subject area, you cannot change its primary object; however, you can create a new custom subject area with a different primary object.

*Related Topics*
- What’s a primary object?

**What happens if I change a custom subject area after it is published?**

You can edit a published custom subject area and then republish it after your changes are done. Modifying a custom subject area does not affect the reports that you created using that custom subject area before making the changes. You can use the modified custom subject area should you need to enhance existing reports.
**Note:** You cannot edit a primary object when you modify a custom subject area. Should you need to do so, create a new custom subject area using a different (new) primary object.
11 Page Customization

Customizing Pages: Overview

Use Page Composer to customize page content, layout, and more. Using other tools, you can create new pages and customize UI text, themes, infolets, and so on.

For example, you can:

- Use tools such as User Interface Text Customization to customize UI text.
- Use the Appearance page to change the look and feel of the application.
- Open an infolet page in Page Composer to customize it.
- Use the Page Integration page to create and manage pages for hosting third party applications.

Page Composer Overview

Prerequisites for Customizing Existing Pages: Explained

Before customizing pages, do the following tasks:

- Understand the typical workflows for working with run time customizations.
- Verify that the page is customizable. To do so, check if either the Customize Pages or the Customize <Page Name> Pages menu item is available under the Administrator menu. If no, then that means the page can’t be customized.
- Confirm that your privileges are sufficient for customizing the page.
- Activate a sandbox.

Related Topics

- Run Time Customization Workflow: Explained
- Customization Layers: Explained
- Setting Up Sandboxes: Procedure

Page Composer Views: Explained

You can use either Design view or Source view for viewing and customizing 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 customization features, you can use some unique features in each view.
Tip: Use the Dock option at the bottom right corner of the page to specify whether the page source appears on top, bottom, left, or right of the Selection pane. This option is available only when you open the page in Source 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 under 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>
</tbody>
</table>
| **Display Options** | Settings that affect the chrome of a component, including:  
  - Header, header text, and border  
  - Actions menu  
  - Edit, Remove, Expand, Collapse, and other icons  
  - Tooltips | For example, display options on image layout components specify the image source URL and its optional link target. |
| **Child Components** | The list of all components contained within and under the control of the parent component, including controls for ordering the child components. |   |
| **Style**        | 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. | 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. |
| **Content Style** | Events and event handlers for all the components on the current page that the current component can consume. | 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. |
| **Events**       |                                                                                                                                               |   |

**Working With Components in Page Customizations: Procedure**

Use the Resource Catalog to customize 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 customizations 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 customize 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>Button &quot;B&quot; appears.</td>
<td>Determines the component that triggers the event.</td>
</tr>
<tr>
<td>What happens when the condition is met?</td>
<td>Button &quot;B&quot; 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</td>
<td>(The implication is that button B is hidden until the event occurs.)</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.

```javascript
#{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

### Customizing Tabs on Application Pages Using Page Composer: Worked Example

This example demonstrates how to customize 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 customization for?</td>
<td>All users</td>
</tr>
<tr>
<td>What customizations are you going to do?</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.

### Customizing Tabs

**Do the following:**

1. From the Navigator menu, select **Tools - Worklist**.
2. Click your user name in the global area, and select **Customize Pages** under Administration.
3. Select **Site layer** as the customization 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 customize.
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 Customizations Visible Based on User Roles: Worked Example

This example demonstrates how you can make customizations 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 customization visible?</td>
<td>Hiring managers, Sherry Callaway and Terrance Whitaker</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> Create a role just for testing customizations. Call it DEVCUST_TEST_ROLE. When you’re sure that the customization works, change the security to the appropriate role.</td>
</tr>
<tr>
<td>Which expression to add for verifying whether a user has the appropriate privilege?</td>
<td>#{securityContext.userGrantedPermission['MANAGERREPORTS LINK_PRIV']}</td>
</tr>
</tbody>
</table>

As a prerequisite, activate a sandbox.

Making Customizations Visible to a User Role

Follow these steps:

1. Create a privilege.
2. Add the 'MANAGER_REPORTS_LINK_PRIV' object to 'DEVCUST_TEST_ROLE'.
3. Assign DEVCUST_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 customize, 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['MANAGERREPORTS_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.
Tip: Use Oracle JDeveloper to define custom permissions that render UI components based on the user’s access rights. (Oracle JDeveloper is not available in Oracle Cloud implementations.)

Related Topics

- Setting Up Sandboxes: Procedure

Customizing Objects That Appear on Multiple Pages: Points to Consider

Use Page Composer to customize objects that appear on multiple pages. Whether your customizations 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 customize objects that appear on multiple pages.

Customization Appearance Based on Shared and Non Shared Task Flows

The following table briefly describes the task flow scenarios when object customizations appear on one or more pages.

<table>
<thead>
<tr>
<th>If the object is...</th>
<th>Then the customizations...</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 customizations 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 customizations, but prevent users from viewing them until the customizations are complete?
Create or select an appropriate sandbox, and set it as active to capture your customizations using Page Composer. When you're ready, publish the sandbox to make your changes available to users.

Related Topics
- Managing Customizations Using Sandboxes: Explained
- Setting Up Sandboxes: Procedure
- Publishing Sandboxes: Procedure

What happens if my customizations or personalizations 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 out-of-the-box state, click Reset Page in Page Composer (Design or Source view).

To reset a task flow to a previously saved version or the original out-of-the-box 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.

How can I use metadata to perform customization-related tasks that I can't do in Page Composer?
In Page Composer, open the Manage Customizations dialog box, and download the metadata to inspect the existing customizations in the metadata.

How can I reset all personalizations made by a specific end user?
Your end users can personalize a page in Oracle Sales Cloud explicitly (like creating a saved search) or implicitly (like changing table column width). You can revert all such personalizations made by a specific user using the Personalization page in Application Composer. In the Common Setup region, click Personalization. Search for a user and then click Reset Personalization. All personalizations made by that user are removed and all pages are restored to the default content and layout as originally delivered by Oracle Sales Cloud. Changes implemented by an administrator using Application Composer and Page Composer are retained, if they exist.

End users can also revert their personalizations themselves, but only on a page-by-page basis.
Related Topics

- How can I restore a page to the default content and layout?

Page Content and Template Customization

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

Customizing 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 customize the global page template. To open the global page template in Page Composer, click your user name in the global area, and select Customize Global Page Template from the Administration menu.

You can make the following customizations 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 customize the template for the desktop UI. Similarly, if you click **Customize Global Page Template** from a simplified page, you will customize the template for the simplified UI. Hence, to ensure a consistent look and feel for all pages, you must customize 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 area 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 area.
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 customize 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 customizations, such as selecting a different color palette, see the Oracle Fusion Applications Extensibility Guide for Developers.

### Customizing Page Titles: Procedure

Page titles appear in multiple places in the application: the browser header, page headings, tab titles, and as items in task lists and the Navigator.

Three separate title and label properties control the different places where the title appears. The properties are:

- Task list Page Title property
- Task list task Label property
- Navigator item node Label property

To customize a page title, you must enter the same value for all three properties. You can change the two task list properties using Page Composer, and the navigator item node property using the Structure page.

**Tip:** You can also make text changes in page titles, task labels, and navigator item nodes using the User Interface Text tool.

If your application requires language localization, you must provide the translated custom text. For more information about translating custom text, see the Oracle Fusion Applications Extensibility Guide for Developers.

### Prerequisites

Activate a sandbox.

### Task List Page Title Property

Use this property to control the text that appears in the browser header.

To change this property:

1. Open the page you want to customize, and then open Page Composer.
2. From the View menu, select **Source**.
3. In the Selection pane, click a task list link. An edit warning confirmation dialog box appears.
4. Select **Do not ask next time**.
5. Click **Edit**.
6. In the Source pane, right-click the `panelFormLayout` node, and select **Edit**. The panelFormLayout Component Properties dialog box appears.
7. Select the **Tasks List Properties** tab.
8. Enter the new value in the Page Title field.
9. Click **Apply** to save the changes, if you are changing the next property now. If you keep the dialog box open, skip the next step and then start at step 6 in the next section.
10. Click **OK** to save the changes and close the Component Properties dialog box.
Task List Task Label Property

Use this property to control the text that appears:

- In page headings and tab titles
- As menu items in task lists

To change this property:

1. Open the page you want to customize, and then open Page Composer in Source view.
2. In the Selection pane, select the task list. A confirmation dialog box appears.
3. Click **Edit** to edit the task flow.
4. In the Source pane, click the **Edit Task Flow** link next to the first subordinate region node.
5. Click the **Edit** icon to edit the task flow. The Component Properties dialog box appears.
6. Click the **Tasks List Task Properties** tab.
7. Expand the tree to display the child items in the task list hierarchy.
8. Right-click the child item that you want to customize and click **Edit** in the toolbar.
9. Enter the new value in the **Label** field.
10. Click **OK** to save the changes and close the Component Properties dialog box.

Navigator Item Node Label Property

Use the Structure page to control the names of categories and page entries that appear as navigator menu items.

To change this property:

1. From the Navigator menu, select **Tools - Structure**.
2. On the Structure page, click the name link for the category or page entry.
   - **Tip:** You can use the search panel on the Structure page to find the category or page entry that you want to rename.
3. On the Edit Page Entry or Edit Category page, enter the new value in the **Name** field.
4. Click **Save and Close**.

**Related Topics**

- Managing Categories and Page Entries for the Navigator and Springboard: Procedure
- Setting Up Sandboxes: Procedure

Customizing Task Lists: Procedures

Use task lists to organize direct links to task flows in your application and web pages outside your application. You link task flows and web pages to tasks through the Tasks List Task Properties in the Component Properties dialog box. Use the Source view of Page Composer to customize task lists.

To customize task lists, you should know how to:

- Insert tasks into the task list.
- Remove tasks from the task list.
- Reorder tasks in the task list.
• Change properties associated with tasks and task lists.

To perform these customizations:

1. Navigate to the task list that you want to change and open Page Composer.
2. If you are in Design view, change to Source view. You customize task lists in Source view only.
3. In the Selection region, click the task list.
4. Click Edit to confirm your intent and automatically select the task list entry in the Source region.
5. In the Source region, click the Edit Task Flow link next to the task list region.
6. Click Edit to confirm your intent and automatically select the task list entry in the Source region.
7. In the Source region, right-click the panelFormLayout node, and select Edit.

The Component Properties: panelFormLayout dialog box appears.
8. Click the Tasks List Task Properties tab.
9. Expand the tree to view the hierarchy of items in the task list.
10. Select an item in the task list hierarchy.
11. The toolbar provides multiple customization options. You can edit the selected item or insert a new task in the task list.
12. If you insert a new task in the task list, you must enter values in several properties.

The following table describes the properties that you use to create a task in your task list.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description or Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Application</td>
<td>Select the target web application from the list of web applications that you defined in the deployments tables. Oracle Cloud customers must contact the help desk at <a href="https://supportoracle.com">https://supportoracle.com</a> to log a service request and obtain a list of valid values.</td>
</tr>
<tr>
<td>Focus View Id</td>
<td>Enter the focusViewId value of the target page, for example, / ServiceRequest Focus View ID and Web Application are mutually inclusive properties.</td>
</tr>
<tr>
<td>Action</td>
<td>Enter the action that occurs when the user selects this item in the task menu. Pages with actions appear in the adfc-config.xml file. These actions can take the user to a particular page. If you specify an action here, then the Web Application and Focus View ID values are ignored. This Action attribute is used in an ADF Controller navigation.</td>
</tr>
<tr>
<td>Label</td>
<td>Enter a name for the new task. This name appears in the task list menu and on the task tab when opened if the value of the page attribute isDynamicTabNavigation is true.</td>
</tr>
<tr>
<td>Rendered</td>
<td>Select this to display the item in the task list. Deselect to hide the item. To help you identify hidden items quickly, deselected items appear in italics on the customization dialog box.</td>
</tr>
<tr>
<td>Destination</td>
<td>Enter the full URL for the item, such as <a href="http://www.example.com">http://www.example.com</a>. Use to open a new window that takes you out of the Oracle Fusion Middleware UI Shell pages. Destination takes precedence over any specified Web Application value.</td>
</tr>
<tr>
<td>Property</td>
<td>Description or Value</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Task Type</td>
<td>Select the task type for newly created items from the list of these values:</td>
</tr>
<tr>
<td></td>
<td>o dynamicMain</td>
</tr>
<tr>
<td></td>
<td>o defaultMain</td>
</tr>
<tr>
<td></td>
<td>o defaultRegional</td>
</tr>
<tr>
<td></td>
<td>o taskCategory</td>
</tr>
</tbody>
</table>

13. Save any changes that you made to the properties, and then save the page in Page Composer.

Customizing Dialog Box Content Using Page Composer: Procedure

Use Page Composer and work in source view to customize the content in your dialog boxes.

Prerequisites

Activate a sandbox.

Customizing Dialog Boxes

To customize 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 customize 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 Customization

Making Saved Searches Available to All Users: Procedure

Use Page Composer to customize 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.

Customizing 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 name in the global area, and select Customize <Page Name> Pages under Administration.
3. If prompted to select a customization layer, select the site layer to open the search page in customization mode.
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 below 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 customize saved searches for other users using Page Composer.

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:
   o Set as Default: The saved search is automatically selected whenever you open this page.
   o 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 customizing the saved search using 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 customize 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 add or reorder filters, if available.
4. Set filter values, 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 custom, 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 customizing the saved search using 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 custom saved searches only, you can also rename or delete the saved search.
6. Click **OK**.

### Simplified Pages Layout and Content

#### Customizing Simplified Pages Using Page Composer: Procedure

On a simplified page, you can customize 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 customizing 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 customizations.

⚠ Note: Any changes you make apply:
- Only to the page you’re on.
- To all or specific groups of users, depending on the customization layer you select before making changes.
Prerequisites
Activate a sandbox.

Customizing a UI Component
To update component properties:

1. Click your user name in the global area and select Customize Pages.
2. Select a customization layer, for example to make changes only for users with a specific job role.

⚠️ Note: When you customize a UI component for a specific job role, that job role must be assigned to you for you to test the customization 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 customize. You can tell you’re in this view when the Design button above 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 above 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 customization 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 customize 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 customizations 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 Customization Layers: Examples
- Setting Up Sandboxes: Procedure
- Customizing the Navigator and Springboard: Overview
- Role Provisioning and Deprovisioning: Explained

### How can I post announcements on the home page?

Use the Announcements page to create, edit, and delete announcements. From the Navigator menu, select **Tools - Announcements**.

- Only the announcement’s content (not subject) appears on the home page.
- If social networking features appear instead of announcements, then on the Set System Options page, change the home panel settings to display announcements. To open this page from the Navigator menu, select **Tools - Structure**, and then click **Set System Options**.
- What you do on the Announcements page applies immediately to all users, even if you saved your changes while a sandbox is active and not yet published.

**Related Topics**
- Setting Up Sandboxes: Procedure
- Defining Settings for Home and Navigation: Explained

### 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 under a single category on the Navigator menu and springboard.

**Prerequisites**

Activate a sandbox.

**Creating the First Page**

Follow these steps:

1. From the Navigator menu, select **Tools - 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 category to place your page under.
5. Search and select an icon for the page.
6. Select the role to whom you want to grant access to the page.
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 customized access, a partner can directly perform an action or display information to the specific user without any additional authentication.

8. To create a partner application for exporting to Oracle Marketplace, do the following:
   a. Select Partner Application for Oracle Marketplace. Once you create the first page without selecting this check box, you can't create partner applications any more.
   b. Enter the extension name and the extension short name for the partner application. The extension short name must start with an alphabet letter or underscore, and contain only alphanumeric characters and underscores.

⚠️ Caution: If the extension short name is invalid, you may destabilize the environment.

9. 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 under a category, then that page icon appears at the top level (not under any category) on the springboard. However, such page icons appear under 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 category as that of the first page, by default.

Related Topics

- Setting Up Sandboxes: Procedure
- Managing Categories and Page Entries for the Navigator and Springboard: 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. Select the application role to whom you want to grant access to the page.
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 under 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**

---

**User Interface Text Customization**
Selecting a Text Customization Tool: 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.

Text customization tools include:

- 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 customization
- The components that you customize

This table presents the navigation and offering availability options associated with the tools you can use to customize 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 <strong>Tools - Application Composer</strong>.</td>
<td>Oracle Sales Cloud</td>
</tr>
<tr>
<td>User Interface Text</td>
<td>In the Navigator, select <strong>Tools - User Interface Text</strong>.</td>
<td>All applications</td>
</tr>
<tr>
<td>Page Composer</td>
<td>Click your user name in the global area, and select <strong>Customize &lt;Page Name&gt; Pages</strong> under Administration.</td>
<td>All applications</td>
</tr>
<tr>
<td></td>
<td>If the Customise &lt;Page Name&gt; Pages option isn't available under the Administration menu, then select <strong>Customize Pages</strong> instead.</td>
<td></td>
</tr>
<tr>
<td>Projects Custom Objects (Application Composer)</td>
<td>In the Navigator, select <strong>Tools - Projects Custom Objects</strong>.</td>
<td>Oracle Project Portfolio Management Cloud</td>
</tr>
</tbody>
</table>

Text Customization Scenarios

The following table includes possible scenarios for customizing user interface text. Compare your situation to the scenario in the table to determine the most appropriate tool for customizing 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.</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. The customization affects multiple messages throughout your application.</td>
<td>User Interface Text</td>
<td>Change the word Employee to Associate 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. The customization affects multiple pages and multiple messages throughout your application.</td>
<td>User Interface Text</td>
<td>Change the word Employee to Associate and Employees to Associates.</td>
</tr>
<tr>
<td>Replace a word or phrase that appears on a specific page</td>
<td>Targeted: A page The customization affects user interface text on a specific page or page fragment.</td>
<td>Page Composer</td>
<td>Change the word Customer to Account 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 The customization affects part of a specific message in the message dictionary.</td>
<td>Manage Messages task</td>
<td>Change the word Recruit to Potential Employee, but only in two specific messages. All other messages continue to use the word Recruit.</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 The customization affects a specific component of a specific message in the message dictionary.</td>
<td>User Interface Text</td>
<td>In Oracle Sales Cloud, change the label of the opportunity business object, from Opportunity to Deal. 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 Total Compensation Statements to Compensation Package Statements.</td>
</tr>
</tbody>
</table>

Regardless of the tool you use to make changes, all customizations are written in a single override bundle. Hence, the latest customization overwrites the previous ones.

**Note:** If you customize using plain text as input, it supersedes any customizations that use the override bundle. For example, if you enter a direct string in Page Composer, then Page Composer writes these customizations to a page customization file (not string resource). Hence, such customizations overwrite the customizations in the override bundle.

**Related Topics**

- Tools for Customizations and Extensions: Critical Choices
Bulk Text Customizations: Explained

Use the User Interface Text tool to simultaneously customize 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 customization:

- 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, from the Navigator menu, select **Tools - 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 customization 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.

Use the sample values in this table as a guide when you enter 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.</td>
</tr>
<tr>
<td></td>
<td>Flex</td>
<td></td>
<td>Unless you select <strong>Match Case</strong>, all matches are considered exact.</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></td>
<td>Flexfields</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

430
<table>
<thead>
<tr>
<th>Search Text</th>
<th>Expected Match</th>
<th>Match?</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<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 Match Complete Word or Phrase 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></td>
<td></td>
<td>Unless you select Match Case, both matches are considered exact.</td>
</tr>
<tr>
<td></td>
<td></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 Match Case, 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>flex credits</td>
<td>flex credit shell plan</td>
<td>No</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 credit for...</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>
The application searches for the exact spelling and sequence of words, where the words appear at the end of the strings it searches.

Unless you select **Match Case**, both matches are considered exact.

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 customization.

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 customization as you see it.
- Select **Exclude** to eliminate the row from the batch customization and maintain the existing text.
- Modify the replacement text to fine-tune the customization for the specific match in the row. The row remains part of the batch customization, even if the actual customization 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 customizations.
2. Thoroughly test the run time pages to make sure that every occurrence of the text is replaced, as you wanted.
3. Publish the sandbox.

Note the following points:

- Do not publish a sandbox before you visually inspect and validate all pages and messages that contain text that you customized.
- Users can view:
  - Message, business intelligence, and enterprise scheduler text customizations when you save them, even if you don’t publish the sandbox.
  - Page text customizations when you publish the sandbox.

Related Topics
- Using Sandboxes: Explained
- Setting Up Sandboxes: Procedure

Adding Translations of Customized Text: Overview
If you install and use multiple languages in your application and you customize text, then enter translations of the customized text for all languages. You can either add translations at run time, or export strings for offline translations. You can enter custom text translations for existing and newly added strings manually at run time.

You can use several customization tools for updating or adding strings. For example, you can use lookups to add translations at run time. 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 Run Time: Worked Example
This example demonstrates how to translate existing strings manually at run time.
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>
</tbody>
</table>
Decisions to Consider | In this Example
--- | ---
What’s your base language? | English
What’s the existing text that you want to customize? | Page
What’s the replacement text that you want to replace the existing text with? | Work area

Entering Custom Text Translations for Existing Strings

1. Activate Sandbox1.
2. From the Navigator menu, select **Tools - 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 customizations 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 Customization Tools: Worked Example

This example demonstrates how to translate new strings that were added using customization tools. While creating strings using customization 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>
</tbody>
</table>
Decisions to Consider | In this Example
--- | ---
What’s the newly created English text that you want to translate in French? | Computer
What’s the French replacement text that you want to replace the newly created English text with? | Ordinateur

Enter Custom Text Translations for Newly Added Strings

1. Activate Sandbox2.
2. From the Navigator menu, select Tools - 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 Customization

Can I undo customizations that I made using the User Interface Text tool if I haven’t published the sandbox?
It depends on types of text customizations in the sandbox. You can undo all text customizations done in the user interface and global menu label by destroying the sandbox before publishing it. However, you can’t undo the text customizations done in messages, analyses and reports, and scheduled processes.

Can I get a report of all customized text if I want to analyze, troubleshoot, and diagnose the cause of unexpected action?
No, but you can use Customization Setup Manager to export all your application customizations to a zip file format. You can find the text customizations in files ending in ".xlf.xml". These files list all text customizations done in your application using
browser-based tools, such as Application Composer, Page Composer, and Customize 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 customizations?
The bulk customizations that you perform using the User Interface Text tool affect only the text that appears on application pages and in message dictionary messages.

Theme Customization

Customizing Themes: Overview

Use themes to customize the look and feel of your application. You can 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 simple theme editor, that is, the Appearance page to create or edit themes.

Prerequisites for Customizing Themes

Before customizing themes:

1. From the Navigator menu, select Tools - Appearance.
2. Activate a sandbox. If you’re not in an active sandbox, click Edit on the Appearance page. 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 page.

Once you complete customizing your themes, you can preview and test your changes, and then publish the sandbox to make your changes available to users.

Customizing Themes

Use the Appearance page to create themes and edit custom themes. For example, you can determine the following look and feel aspects of your application pages:

- Logo
- Background image
- Size and style of icons on the springboard
- Style of the cards, which appear on a page in a grid view. These cards present 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
When you edit your theme using the Appearance page, the themes for both desktop UI and simplified UI are edited simultaneously.

Apart from the editing options available on the Appearance page, if you need advanced controls to edit your themes, then use the advanced theme editor.

⚠️ **Caution:** Once you edit a theme using the advanced theme editor, you can’t edit that theme using the Appearance page anymore. Also, using the advanced editor, you must separately edit your theme twice - once for the desktop UI and again for the simplified UI.

To edit a theme using the advanced editor, on the Appearance page, select **Advanced** from the Actions menu, and then select either of the following:

- **Edit Desktop UI - <Theme Name>**
- **Edit Simplified UI - <Theme Name>**

**Related Topics**
- **Setting Up Sandboxes: Procedure**

---

**Creating Themes: Procedure**

Create your own custom themes using the simple theme editor, that is, the Appearance page.

**Prerequisites**

Activate a sandbox.

**Creating Themes Using the Simple Theme Editor**

Follow these steps:

1. From the Navigator menu, select **Tools - Appearance**.
2. On the Appearance page, select a theme as your base theme from the Appearance list.
3. Make changes to the theme, for example, change the logo, background, style of icons, and heading colors.
4. From the Actions menu, select **Save As**.
5. On the Save As dialog box, enter a theme name without any spaces.
6. Optionally, select **Apply this theme**.
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 set as the current theme.

When you create a theme using the simple editor, two new themes are saved - one for the desktop UI and another for the simplified UI.

**Related Topics**
- **Setting Up Sandboxes: Procedure**
Managing Themes: Procedure

Use the Appearance page to manage your themes. To open the Appearance page, from the Navigator menu, select **Tools - Appearance**.

You can edit custom themes, apply themes to your application pages, and delete custom themes. You can’t edit or delete any predefined theme.

**Prerequisites**
Activate a sandbox.

**Applying Themes**
Select a theme from the Theme list, 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**
From the Actions menu, select **Apply Default**. The default theme is applied to your application.

**Deleting Themes**
From the Theme list, select a custom theme that you want to delete, and then select **Delete** from the Actions menu.

**Editing Themes**
Follow these steps:

1. From the Theme list, select a custom theme that you want to edit.
2. Make changes to the theme specifications, for example, change the logo, background image, icon style, and color schemes, such as heading and button colors.
3. Click **Apply**.

**Note:** Although you can’t edit a predefined theme, you can save it as a new theme, and then edit and apply it to your application. Use the **Save As** option from the Actions menu to save a predefined theme as a new theme.

**Changing the Logo and Background Watermark**
Use the Appearance page to define the following:

- Branding logo, which appears above all application pages. Use an image that’s as close to 119 by 25 pixels as possible. In general, an image that’s wider than it’s tall works best.
- Watermark, which appears in the background of all simplified pages. Use an image that’s as close to 1024 by 768 pixels as possible.

You can use any of the following options to select a logo and a background image:

- **File**: Browse and select a file from your local machine.
- **Predefined**: Select a file from the list of predefined images.
• **URL**: Enter a full URL for the logo or the watermark.

## Editing Themes Using the Advanced Theme Editor

Use the Theme Editor page only if you want to edit your themes using advanced controls, which are not available on the Appearance page. Unlike the simple editor, you can’t use the advanced editor to edit themes for the desktop UI and the simplified UI simultaneously. So, to ensure a consistent look and feel of your application across these UIs, you must edit each theme separately for each UI.

⚠️ **Caution**: Once you edit a theme using the advanced editor, you can’t edit it using the simple editor anymore. You must use only the advanced editor for subsequent edits to that theme.

Follow these steps:

1. On the Appearance page, select the custom theme that you want to edit using the advanced editor, from the Theme list.
2. Click **Advanced** from the Actions menu, and then select either of the following options:
   - **Edit Desktop UI - <Theme Name>** to edit the theme for the desktop UI
   - **Edit Simplified UI - <Theme Name>** to edit the theme for the simplified UI

   The theme opens in the advanced theme editor.
3. Edit the theme, for example, make changes to the branding area, buttons, links, and tabs.
4. Click **Save**.

### Related Topics

• Setting Up Sandboxes: Procedure

## Changing the Logo and Color Schemes of the Application: Worked Example

This example demonstrates how to change a company logo and the color schemes of an application using the Appearance page. Users see the logo in the global area.

📝 **Note**: If the company logo is changed using Page Composer, then this overwrites the changes done using the Appearance page.

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 image are you going to use as the new logo?</td>
<td>MyCompany.gif</td>
</tr>
</tbody>
</table>
Tip: The recommended image size of the logo to be used for any theme is 119x25 px.

Prerequisites
Activate a sandbox.

Changing the Logo and Color Schemes
1. On the Appearance page, select the Default theme from the Theme list.
2. From the Logo list, select File as the type of location where your logo is stored.
4. For the icons, select the style as Flat Dark, the size as Large, and the corner rounding value as 5.
5. In the Color Scheme section, 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>Background Color</td>
<td>A9A9A9 (that is, Dark Gray)</td>
</tr>
<tr>
<td>Global Region Label Color</td>
<td>A52A2A (that is, Brown)</td>
</tr>
<tr>
<td>Global Region Background Color</td>
<td>FFFFFFF (that is, White)</td>
</tr>
<tr>
<td>Heading Color</td>
<td>8B008B (that is, Dark Magenta)</td>
</tr>
<tr>
<td>Page Color Link</td>
<td>0000FF (that is, Blue)</td>
</tr>
<tr>
<td>Button Colors</td>
<td></td>
</tr>
<tr>
<td>Label: A52A2A (that is, Brown)</td>
<td></td>
</tr>
<tr>
<td>Border: 000000 (that is, Black)</td>
<td></td>
</tr>
<tr>
<td>Background: FFA07A (that is, Light Salmon)</td>
<td></td>
</tr>
<tr>
<td>Card Style</td>
<td>Light</td>
</tr>
</tbody>
</table>

6. From the Actions menu, select Save As.
7. On the Save As dialog box, enter the theme name as MyCompany.
8. Select Apply this theme.
9. Click OK.
   The following two new themes are saved in the repository:
   - MyCompanyalta for the simplified UI
   - MyCompanyskyros for the desktop UI

Related Topics
- Setting Up Sandboxes: Procedure
FAQs for Theme Customization

What happens to my theme if changes that affect themes are made using Page Composer?
Customizations made using Page Composer overwrite the changes made using the simple or the advanced theme editor.

⚠️ Caution: Before using theme editor to change the look of your application, you must undo any customizations affecting the theme done using Page Composer. Otherwise, the customizations that you make using theme editor may not show up in your application as you wanted.

Infolet Customization

Customizing Infolets: Overview
Use infolets to aggregate key information for a specific area, for example, your sales pipeline, employee turnover, or other business transactions. You can click the navigation dots on the home page to open infolets and view important information at a glance.

If the navigation dots aren’t available on the home page, you can enable infolets using the system options on the Structure page.

Customizing Infolets
You can open the infolets page in customization mode and perform the following customization 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 Infolets

You can enable infolets, that is, display the navigation dots on the home page using Set System Options on the Structure page.

Related Topics

- Personalizing Infolets: Procedure

Creating Infolets: Procedure

Use the infolet page to create infolets and set their views.

Prerequisites

You must use the infolet page in customization mode, that is, open the infolet page in Page Composer.

Follow these steps:

1. Activate a sandbox.
2. Click a navigation dot on the home page to open the infolet page.
3. On the infolet page, click your user name in the global area, and select Customize Pages under Administration.
4. Select a customization layer. The infolet page opens in customization mode.

⚠️ Note: Once you complete your customizations, 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.
4. Click Save and Close.

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 use the infolet page in customization mode, that is, open the infolet page in Page Composer.

Follow these steps:

1. Activate a sandbox.
2. Click a navigation dot on the home page to open the infolet page.
3. On the infolet page, click your user name in the global area, and select Customize Pages under Administration.
4. Select a customization layer. The infolet page opens in customization mode.

Note: Once you complete your customizations, 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
Follow these steps:

1. Click Add Content on the infolet. This button is available on an infolet only if the infolet doesn’t have any content in it.
2. 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.

Editing Infolet Content
You can edit the tile content of an infolet.

Follow these steps:

1. Click the Actions icon on the top right corner of the infolet, and select Edit Content.
2. Click Add Content to replace the existing content of the infolet.
3. 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. 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**.

### Related Topics

- Setting Up Sandboxes: Procedure
- Personalizing Infolets: Procedure

### FAQs for Infollet Customization

#### How can I enable infolets?

Follow these steps:

1. Select **Tools - Structure** from the Navigator menu.
2. Click **Edit**.
3. On the Sandbox Required dialog box, click **Activate Sandbox**.
4. Select a sandbox and set it as active.
5. On the Structure page, click **Set System Options**.
6. Select **Enable Infolets**.
7. Click **Save and Close**.

✏ **Note:** To make the settings changes available to users, publish the sandbox.

#### 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.

### Design Time Page Customization

#### Customizing Pages Using Design Time Tools: Highlights

Developers can customize pages at design time, for example, to change the color theme for all pages. These changes include complex ones that you need to deploy into the run time environment.
Note: Design time customizations and extensions aren’t available in Oracle Cloud implementations.

Design Time Tools

- Use Oracle Application Development Framework (Oracle ADF) Skin Editor to change the look and feel of your application. Refer to the Oracle Fusion Applications Extensibility Guide for Developers.

  See: Customizing the Oracle Fusion Applications Skin

- Edit the UI Shell template in Oracle JDeveloper to do customizations that you can’t do using Page Composer. Refer to the:
  - Oracle Fusion Applications Extensibility Guide for Developers
    See: Editing the UI Shell Template
  - Oracle Fusion Applications Developer’s Guide
    See: Introduction to Implementing the UI Shell

Customizing the Same Page with Multiple Tools: Points to Consider

Using both Oracle JDeveloper and Page Composer, design time customizations and run time customizations can coexist on a page.

Note: Design time customizations aren’t available in Oracle Cloud implementations.

When you use multiple tools to edit the same page, consider these points:

- You can’t use both tools at the same customization layer.
- If you customize the same component with both tools, the customization at the higher level layer takes precedence.

For example, suppose you use Page Composer to change a field label from Employee to Associate in the site layer. Meanwhile, someone uses Oracle JDeveloper to change the same label from Employee to Worker in the global layer. The global layer is the base customization layer, as it’s only for design time customizations and applies to all users. Your users see Associate, not Worker, because the site layer is at a higher level than the global layer.

- Run time customizations aren’t automatically visible in Oracle JDeveloper. To view them, you must:
  - Export the customizations from the run time environment to a Java archive (JAR) file.
  - Import the customizations into the Oracle JDeveloper customization application workspace.

For more information about importing run time customizations into Oracle JDeveloper, see the Oracle Fusion Applications Extensibility Guide for Developers.

Related Topics

- Customization Layers: Explained
Using Page Composer in Oracle Sales Cloud

Overview

Customize 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 customize pages
- How to customize standard desktop pages, landing pages, and dashboards
- How to customize simplified pages
- How to use Direct Selection mode to make user interface changes
- How to work with components for customizing pages

Note: Page Composer supports two editing modes: Design View and Source View. In Oracle Sales Cloud, customizations can be done only in Design View mode. The only exception is the Partner Relationship Management’s Partner Portal UI Shell customization, which is done in Source View mode.

Page Composer was also known as Oracle Composer in previous releases.

Related Topics
- Customizing Simplified Pages Using Page Composer: Procedure

Customizing Oracle Sales Cloud UIs: Points to Consider

You can use multiple tools to customize Oracle Sales Cloud user interfaces. This topic highlights the differences between the types of customizations 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 customizations 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>Customizing the page you are viewing</td>
<td>Customize 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.</td>
<td>Because Application Composer customizes application objects, making a field required affects all interfaces where that field appears.</td>
</tr>
<tr>
<td>Key Differences</td>
<td>Page Composer</td>
<td>Application Composer</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>page, the field is not automatically required on the Create Opportunity page.</td>
<td>Cannot customize dashboard pages.</td>
</tr>
<tr>
<td>Customizing dashboards</td>
<td>Configure the format and content of dashboard pages. Dashboards can display BI Analytics and BI Publisher reports, notifications, as well external content from your intranet or from the Internet.</td>
<td>Cannot customize dashboard pages.</td>
</tr>
</tbody>
</table>
| Supported interfaces                                | • Simplified UI  
• Desktop UI                                                                                                                                                                                         | • Simplified UI  
• Desktop UI  
• Mobile  
• Microsoft Outlook                                                                                                                                 |
| Creating custom fields and application objects      | Not supported.                                                                                                                                                                                               | You can add custom fields and entirely new objects.                                                                                         |
| Creating saved lists                                | Create and edit saved lists (also called saved searches) for users. The saved lists provide saved search criteria to speed up common searches.                                                              | Not supported.                                                                                                                              |
| Tailoring your customizations for users with different job roles | Provides different levels of customizations for both simplified UI and desktop UI.                                                                                                                             | Supports customization by job role for the simplified UI only. Customizations in other UIs apply to all users.                                 |
| Customizing external-facing partner pages           | You must use Page Composer to customize all external-facing partner pages, such the partner registration page and the partner portal landing page.                                                            | Only for partner pages internal to your organization, such as the partner Overview page.                                                    |

### 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 Manage Menu Customizations task from the Setup and Maintenance work area.
- Customize the Search region inside Search and Select dialog.
Can I use Page Composer to customize all elements on a page?

No, you cannot use Page Composer to customize 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 for customization in the Component Properties window.

⚠️ **Caution:** Do not edit the **Taskflow Id** parameter in Page Composer Task List properties. Doing so may corrupt page metadata.

In general, you can customize 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
• fnf:applicationsTable
• af:commandMenuItem
• af:menu
• af:panelGroupLayout
• af:panelTabbed
• af:group
• af:commandButton

Customizing Oracle Sales Cloud Pages: Explained

You can use Oracle Page Composer to customize both desktop and simplified user interface (UI) pages while you are working in the application. You can make your UI customizations 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 should use Page Composer, also known as Oracle Composer, for performing the following types of customizations:

• 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 customize the UI for a specific job role, then you must have that job role assigned to you so that you can test the customization in the sandbox.
Accessing Page Composer
Accessing Page Composer differs slightly in the simplified and desktop UIs:

- In the simplified UI, click on the user name and select **Customize User Interface**.
In the desktop UI, select the first customize action under the Administration heading. The action name changes with the work area. For example, to customize opportunity pages, you select Customize Opportunities Pages.

Page Composer Customization Modes

In Oracle Sales Cloud, you can use Page Composer in one of two modes: the Design mode and the Select mode.

- Use the Design mode to move, add, and edit regions on dashboards and external-facing partner pagers. You also use this mode also to create saved searches, specify search result content, and to navigate in the UI when you are in Page Composer.

You are in the design mode when you first open Page Composer. You can also select this mode by clicking Design on the top left corner of a page.
• Use the Select mode, which you can activate by clicking Select, to customize the properties of different UI elements such as fields and table columns.

Working in the Design Mode

Use the Design mode to:

• Customize 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

• Customize all of the external-facing landing and transactional partner management pages

• Customize the table display of search results

• Create saved searches (saved lists)

• Navigate to other pages while working in Page Composer
When you launch Page Composer, the application opens in the Design mode automatically. On pages that support customization in this mode, such as the sales dashboard in the following image, different regions are highlighted with borders and display customization controls. Numbers are added to highlight different features of the Design mode.

- A bar appears at the top of the page indicating that you are in editing the page in Page Composer (1).
- You can change the column layout of the page by clicking **Change Layout** (2).
- You can add a new tab (3).
- You can move a region, by grabbing its toolbar (4) and dragging it to a different position.
- You can remove a region by clicking the close 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 on one of the icons to the right of the Add Content button. These icons specify where in the page the new region will be created.

Working in the Select Mode
In the Select mode, a blue 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 customization displays a menu with one or both of the following options: **Edit Component** and **Edit Parent Component**.

Not all UI elements support customization, so these editing options are not always enabled.

You use the Component Properties window to make your customizations. Not all the customizations are available on all the UI elements, so the content of this window varies. The numbers in the following image refer to the customizations descriptions.
To make a field read-only, you select the **Read Only** option (1).

To make a field required, you 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, you deselect the **Show Component** option (3).

To change the field label:

a. Click the down arrow to the right of the **Label** field (4)
b. Select **Select Text Resource** from the menu.

The Select Text Resource window appears.
c. Enter a name in the **Key** field. You can use this name to search for this label when you customize 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.

---

**Editing the Partner Portal UI Shell Using the Page Composer Source View**

Channel partner portal administrators can use Page Composer’s Source View to customize 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 customize 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.

Making Customizations Available to Different User Groups

You can make your customizations available to different subsets of users.

When you launch Page Composer, you must select one of the following customization layers:

- **Site**: Your customizations are visible to all users.
• **External or Internal**: Depending on your selection, your customizations are visible to either internal users (your employees) or to external users (your partners or anonymous users).

• **Job Role**: Your customizations are visible to users with the job role you select.

> **Note**: If you created custom job roles, you must run the Retrieve Latest LDAP Changes process from the Scheduled Processes work area before the custom job roles are available for selection.

By default, customizations 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 customization layers. You must customize this page at the site level.

**Related Topics**

• Customizing What Type of Information Displays in the Desktop UI for a Customer: Worked Example

• Customizing Which Columns Display in Desktop UI Search Results: Worked Example

• What’s Required for Testing Customizations in the Sandbox

## Customizing Simplified Pages: Procedure

On a simplified page, you can customize 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 customizing 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 customizations.

**Note:** Any changes you make apply:
- Only to the page you’re on.
- To all or specific groups of users, depending on the customization layer you select before making changes.

Prerequisites
Activate a sandbox.

Customizing a UI Component
To update component properties:

1. Click your user name in the global area and select **Customize Pages**.
2. Select a customization layer, for example to make changes only for users with a specific job role.

**Note:** When you customize a UI component for a specific job role, that job role must be assigned to you for you to test the customization 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 customize. You can tell you’re in this view when the Design button above 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 above 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 customization 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>
</tbody>
</table>
### Property | Description
--- | ---
Short Desc | Text that appears when users hover or focus on the component, for example hover over a field label or click in the text box.
Show Component | Whether the component is visible or hidden to users.
Show Required | Whether an asterisk is displayed to indicate that the component is required.

8. To customize 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 customizations 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 Customization Layers: Examples
- Setting Up Sandboxes: Procedure
- Customizing the Navigator and Springboard: Overview
- Role Provisioning and Deprovisioning: Explained

### Personalizing Dashboards, Transactional Pages, and Landing Pages: Overview

This topic provides an overview of how you customize your dashboards, transactional pages, and landing pages using page composer.

These customizations are available only to those users who customize them. If you customize your dashboard, only you can see the customized dashboard.

To access the personalization tasks available:

1. Click on your user name in the global region.
2. Select **Settings and Actions - Edit Current Page**, which is under Personalization category.

You perform your personalizations in Design Edit mode. Here’s an overview of what you can personalize in your pages:

- Change the local area layout. For example, you can change a two-column layout to a three-column layout. In all, eight layouts are available for selection.
- Add, rename, or remove subtabs.

> **Note:** You cannot rename or remove predefined tabs or home tabs.
• Expand or collapse the dashboard regional pane. You can do this by moving the page splitter location.
• Add or remove panel boxes from the dashboard local area.
• Add Resource Library content to the dashboard panel.
• Edit the properties of dashboard panel box to show or hide a box, reorder child regions, or change the display and style options.
• Add, remove, or edit dashboard ADF components, such as regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

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

Making Columns Unavailable for Display in Desktop UI Search Results: Worked Example

Use this example to learn how to use Page Composer to make columns unavailable for display in desktop UI search results. When you make the columns unavailable for display, the no longer appear in the table or the View menu and so are completely hidden from users.

Before you customize 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 customizations 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 customizations separate from other customizations, 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.

### Making Columns Unavailable for Display in Search Results

This example shows you how to customize the table of search results in the Customers work area.

1. Select **Customers** in the Navigator.
   The Overview page appears.
2. Click on the user name and select **Customize Customers Pages** from the menu.
3. In the Customize Customers Pages dialog box, specify the set of users that will be affected by this customization by choosing one of the following:
   a. **Site** to make the customization available to all users
   b. **External or Internal** to make the customization available to either internal (employee) users, or to external users such as partners.
Job Role to make the customization available to users with a job role you specify.

4. Click OK.

The page opens in the Page Composer Design mode.

5. Click Select.

6. Mouse over any of the column headers until the blue border appears and click the column header.

A window appears with two options: Edit Component and Edit Parent Component.

7. Click Edit Parent Component.

The Component Properties window appears.
8. Select the **Children** tab.

![Component Properties: resId1](image)

9. Deselect the check boxes for the columns you want to remove from the table. You can also use the arrows on the right to rearrange the columns.

10. Click **OK**.

   The columns you deselected are no longer present in the search table or the View menu which lists the additional columns available for display.

11. Remember that your changes are not permanent until you publish the sandbox.

### Customizing Search Results Information: Worked Example

You can use Oracle Page Composer to customize the search results table to display the information you want users to see. For example, you may want salespersons to see different information than partner managers.

The following worked example shows you how to add information columns to the search results table.

#### Customizing the Search Results Table

1. Sign in as a user with the Sales Administrator job role and activate a sandbox.
2. Select Navigator, and, under the Tools heading, select **Customization**, then **Application Composer**.
3. Select either **Custom Objects** or **Standard Objects**, depending on which object you want to edit.
4. Select the desired object.
5. Select **Pages**, and click the **Edit Summary Table for [Object]** link.
6. Under Configure Summary Table, select the fields you want to appear as columns in the search results table, and click **Save and Close**.
7. Expand the drop-down arrow next to your user name and select **Customize User Interface**.
8. On the Customize User Interface window, select **Job Role** as the Layer, select the job role that you want to see the customized search results information, and click **OK**.
9. Navigate to the page you want to edit and click **Select** on the upper left of the page to enter Select mode.
10. Find the column you want to edit, hover over the column header you want to change, and click **Edit Parent Component**.
11. On the Component Properties window, select **Children**.
12. On Component Properties, select to show or deselect to hide any column.
13. Click **OK**.
14. On the upper right on the page, click **Close** to close Oracle Page Composer.
15. Navigate to the list page and confirm that the correct information appears.

### Note:
You must have the appropriate job role assigned to you to see the changes.

16. Publish the sandbox.

**Related Topics**

- Assigning Yourself Additional Job Roles Required for Testing

---

**Customizing 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 customize the global page template. To open the global page template in Page Composer, click your user name in the global area, and select **Customize Global Page Template** from the Administration menu.

You can make the following customizations 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 customize the template for the desktop UI. Similarly, if you click **Customize Global Page Template** from a simplified page, you will customize the template for the
simplified UI. Hence, to ensure a consistent look and feel for all pages, you must customize 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 area 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 area.
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.
3. When you move your cursor over the global page template, the areas that you can edit display a blue outline.
4. Click Delete. When prompted, click Delete to delete the component.
5. 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 customize 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 customizations, 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 customizations are preserved in the active sandbox, so there is no loss of data. Page Composer sometimes quits because it was designed to customize one object at a time.
Chapter 13
Understanding Analytics and Reports

Customizing Sales Cloud Analytics and Reports: Overview

Oracle Sales Cloud comes predefined with a wealth of business intelligence (BI) analytics and reports that help your sales team monitor and interpret your sales pipeline, team performance, forecasts, activities, customers, and more. If the supplied analytics and reports don’t meet your unique business requirements, you can build your own.

This topic provides an overview of customizing analytics and reports. For detailed information, see the online help and the Oracle Sales Cloud Creating and Administering Analytics guide.

Build Custom Analytics and Reports

You use the BI presentation catalog to build and view custom analytics and reports. You can browse the subject areas, which are organized by functional area, such as opportunities, leads, and the like. You can also create custom subject areas. Within each subject area are columns and attributes that you use to create the reports. Each report can have its own layout and format, such as table format or graph format. You can also filter attributes by various criteria.

You access the BI presentation catalog from the Navigator. Click Reports and Analytics in the navigator Tools menu.

Customizing Infolets

The Sales dashboard in the simplified UI comes predefined with several infolets, which are configurable portals that provide report summaries based on transactional sales data. Administrators can create custom infolets and display them in the Sales area of the simplified UI.

Related Topics

- Oracle Help Center
14 Navigator and Springboard Customization

Customizing the Navigator and Springboard: Overview

You can customize the Navigator and springboard, as well as define settings for the home page and springboard using the Structure page. To address needs specific to your organization, you can create or edit categories and page entries for the Navigator and springboard. For example, you may want to link page entries to web pages or external applications.

Categories and Page Entries

Categories and page entries are the navigator menu items. 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 grouped under a category. Depending on the number of page entries that you have access to, the page entries can appear at the top level (not in any category folder) on the springboard. If you have only one page entry under a category, then that page entry icon appears at the top level (not under any category) on the springboard. However, such page entry icons appear under their respective categories on the Navigator menu.

While creating or editing a page entry or category, you can use the Show on Welcome Springboard field to specify whether to display them on the springboard. So, not all page entries and categories may appear on the springboard. The Navigator menu may have more page entries and categories 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.

Customizing the Navigator and Springboard Using the Structure Page

For customizing the Navigator and springboard, use the Structure page to do the following tasks on categories and page entries:

- Create
- Edit
- Show or hide
- 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.

Prerequisites for Customizing the Navigator and Springboard

Before customizing the Navigator and springboard using the Structure page:

1. From the Navigator menu, select Tools - Structure.
2. Activate a sandbox. If you're not in an active sandbox, click **Edit** on the Structure page. You're prompted to activate a sandbox.

💡 **Tip:** If you're already in an active sandbox, then the Edit button doesn't appear on the Structure page.

If prompted, select a customization layer to determine the scope of users that your changes affect.

**Related Topics**
- Setting Up Sandboxes: Procedure
- Navigating in the Application: Explained
- Setting Profile Option Values: Procedure

**Customizing the Navigator and Springboard**

💡 **Watch:** This tutorial gives you an overview of the Navigator and springboard and shows you how to customize their categories and page entries.

**Creating Categories and Page Entries for Navigation:**

**Procedure**

Use the Structure page to create categories and page entries for customizing the Navigator and springboard. From the Navigator menu, select **Tools - Structure**.

You can do either of the following:

- Create a category and then create a page entry in that category.
- Create a page entry in an existing category or at the top level (not in any category).

**Creating Categories and Page Entries**

To create a category or a page entry:

1. Click **Create**, and select **Create Category** or **Create Page Entry**.
2. Enter a name for the category or page entry. For a category, the one available icon is already selected for you.
3. If you're creating a page entry:
   - Search and select an icon for the page entry.
   - Select the category in which you want to place the new page entry.
4. Select **Yes**, **No**, or **EL Expression** in the Visible field:
   - Yes: The category or page entry appears on the Navigator. It can also appear on the springboard, depending on what you select in step 6.
   - No: The category or page entry doesn't appear on the Navigator and springboard.
EL expression: The evaluation of the EL expression decides whether the category or page entry will appear on the Navigator and springboard.

5. If you have selected EL Expression for the Visible field, click the Edit icon next to the Visible drop-down list, and enter a value or expression. Don’t include spaces or double quotes in the EL expression.

For example, depending on the user role that you want to display the categories or page entries for, enter the EL expression as described in the following table:

<table>
<thead>
<tr>
<th>Who can see the category or page entry</th>
<th>EL Expression and Example</th>
</tr>
</thead>
</table>
| Only users having any of the specific roles | #{securityContext. userInRole['<RoleName>']}
| | #{securityContext. userInRole['ORAFND APPLICATION_ADMINISTRATOR_JOBORA_PER_EMPLOYEE_ABSTRACT']} |
| Only users not having any of the specific roles | #{!(securityContext. userInRole['<RoleName>'])}
| | #{!(securityContext. userInRole['ORAFND APPLICATION_ADMINISTRATOR_JOBORA_PER_EMPLOYEE_ABSTRACT'])} |
| Only users having all of the specific roles | #{securityContext. userInAllRoles['<RoleName>']}
| | #{securityContext. userInAllRoles['ORAFND APPLICATION_ADMINISTRATOR_JOBORA_PER_EMPLOYEE_ABSTRACT']} |
| Only users not having all of the specific roles | #{!(securityContext. userInAllRoles['<RoleName>'])}
| | #{!(securityContext. userInAllRoles['ORAFND APPLICATION_ADMINISTRATOR_JOBORA_PER_EMPLOYEE_ABSTRACT'])} |

**Note:** Categories and page entries are evaluated on all pages. So roles used in the EL expression for the Visible field must be available in all application stripes, that is, in CRM, FSCM, and HCM.

6. For a page entry, if you have set the Visible field to Yes or EL Expression, then set the Show on Welcome Springboard field. The value of this field is evaluated to determine if the page entry will actually appear on the springboard.

   o Yes: The page entry appears on the springboard.

   Suppose you have specified an EL expression in the Visible field. Then, even on setting the Show on Welcome Springboard field to Yes, the page entry may not appear on the springboard. The evaluation of the EL expression decides whether these items will actually appear on the springboard.

   o No: The page entry doesn’t appear on the springboard.

   o EL expression: The evaluation of the EL expression decides whether the page entry will appear on the springboard.

7. If you’re creating a category, then click **Save and Close**. If you’re creating a page entry, then perform steps 8 to 10 instead of this step.

8. Select any of the following link types for the page entry:

   o 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.

9. Based on the link type, specify the required details to configure the link.
10. Click **Save and Close**.

## Configuring Links for Page Entries: Procedure

While creating a page entry or editing a custom page entry for the Navigator and springboard using the Structure page, 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 for enhanced security.

### Linking to Application Pages

This option is applicable for non-Cloud implementations only. For Oracle Cloud implementations, you can configure the links to application pages while creating your pages using the Page Integration Wizard page.

To link a page entry to one of your application pages:

1. Select the Application Page link type.

   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.

2. Enter the focus view ID of the target page.

   You can get the Focus View ID from the value of the page’s `view_id` attribute in the product’s `public_html/WEB-INF/adfc-config.xml` file.

3. Select the name of the web application.

   This is the application name that you had entered while creating this third party application using the Setup and Maintenance work area.

4. For a 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 category on the springboard or Navigator, the parameter might be set to `status=Open`. Whereas, if you’re opening the page from another category, the parameter might be set to `status=Closed`.

**Linking to Dynamic URLs**

You can link a page entry to an external web site 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, say you need 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 will affect 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, say, you need to link to a complete URL: http://example:9011/myApp/faces/Page1.

Then:

- The name of the web application added to topology will be: `myApp` (the value that would eventually appear in the web application list). The protocol host, port, and context root values of the URL will be: http://example:9011/myApp.
- The destination for the web application will be: `/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 web site 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. Specify the URL destination.

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. A HTTPS protocol is required to access the application.
5. Enter the name of the secure token. The secure token expires if the user session is inactive. So, users need to 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 customized access, a partner can directly perform an action or display information to the specific user without any additional authentication.

Related Topics
• Creating Pages for Hosting Third Party Applications: Procedure

Managing Categories and Page Entries for the Navigator and Springboard: Procedure

Use the Structure page to edit and reorder the existing categories and page entries for customizing the Navigator and springboard.

Editing Categories and Page Entries

Follow these steps:

1. From the Navigator menu, select **Tools - Structure**
2. On the Structure page, click the name link for the category or page entry.

   **Tip:** You can use the search panel on the Structure page to find the category or page entry you want to edit.

   If you get a "webApp value not defined" error message on clicking a category or page entry, verify whether the application is in the topology tables. For more information, refer to the Oracle Fusion Applications Administrator's Guide.

3. On the Edit Page Entry page or the Edit Category page, make the required changes.
4. Click **Save and Close**.

You can make the following changes to a category or page entry:

• Rename a category or page entry.

   **Note:** If a category or page entry was created using a different tool, then you can’t change its name using the Structure page.

• Change the icon for a page entry.
  - You can’t change the icon for a category.
  - If a page entry was created using a different tool, then you can’t change its icon using the Structure page.

• For a page entry, change the category under which the page entry is grouped.
• Change the Visible property for the category or page entry.
• For a custom page entry, change the settings for link configuration.
• Delete page entries that are created using the Structure page.

Editing Page Entries with Tabs
Some pages (for example, Security Console) have tabs. Each tab is a task flow. The tabs are displayed at the bottom of the Edit Page Entry page. On the Edit Page Entry page, you can:

• 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

Reordering Categories and Page Entries
Use the **Move Up** and **Move Down** icons on the Structure page to reorder categories and page entries. For page entries, you can use the **Move To** icon to move page entries to different categories or to the top level.

Related Topics
• Customizing Themes: Overview

Defining Settings for Home and Navigation: Explained
Click **Set System Options** on the Structure page to define settings for the home page and springboard. To open the Structure page, from the Navigator menu, select **Tools - Structure**.

**Note:** You must be in an active sandbox to define these settings.

Defining Settings
You can define the following settings:

• **Home Panel**: Specify the content of the side panel that appears to the left of the springboard on the home page. Select to display either social networking content or announcements in the panel. By default, social networking content is displayed in the home panel. Also, this panel isn't visible to partners or other external users who sign in to your application, by default. To provide access to such users, contact your security administrator.

• **Enable Infolets**: Select the infolet pages that you want to include on the home page. Based on your selection, these infolet pages will appear as a row of dots on the home page. Users can click a dot to navigate to each included infolet page. If you don’t select any infolet page in the settings, no such dots will appear on the home page.

You can use profile options to define settings for the springboard strip that you can find above all simplified pages:

• To enable users to use the springboard strip, 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 springboard strip 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 once done, this profile option is set by default for subsequent sessions of that user.

Related Topics
• Setting Profile Option Values: Procedure

FAQs for Navigator and Springboard Customization

Why can't I edit the Structure page entry or the Tools category?
While customizing the Navigator and springboard, you can't:
• Override the Visible setting for the Structure page entry and Tools category. The default setting is Yes, so the Structure page entry and Tools category always appear on the Navigator and springboard.
• Move the Structure page entry to a different category or to the top level. The default category is Tools, so the Structure page entry always appear under the Tools category.

Why are some springboard icons not displayed on my springboard?
A springboard icon, that is, a group or a page entry may be hidden due to any of the following reasons:
• You may not have security privileges to access or view the group or page entry. To review, check if the group or page entry is available on the Navigator menu. If no, then that means you don't have the appropriate privileges.
• The administrator has hidden the group or page entry from the springboard using the Structure page. To review the visibility settings, from the Navigator menu, select Tools - Structure.
• The administrator hasn't enabled the offering associated with the group or page entry.
• The accessibility preference for the application is set to the screen reader mode. To review accessibility preferences, in the global area, either click the Accessibility icon or select Settings and Actions - Personalization - Set Preferences.

Related Topics
• Enabling Offerings: Explained
• Enabling Offerings: Procedure
• Accessibility Preferences: Explained
• Why are springboard icons appearing at the top level instead of under a folder?
15 Help Customization

Customizing Help: Overview

You can customize help files in Applications Help and help windows, as well as help text that appears on UI elements. The Getting Started work area can also be customized.

Note: To enable help file and Getting Started customization, you or your implementor must select the Help Customization feature choice in your offering configurations.

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 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 name in the global area and select Applications Help.

You have many options to customize 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 customize predefined pages and create pages with your own custom 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 customize 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

- **Features: Explained**
Who can add and manage custom help?

Users with the Customize Help Topics (ATK_CUSTOMIZE_HELP_TOPICS_PRIV) privilege can customize:

- 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 File Customization

Overview

If you have the appropriate roles, then you can customize the 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 tabs a file belongs to in Applications Help.

What you can do to a help file depends on whether it’s custom or predefined.

- **Custom:**
  - Create, duplicate, edit, and delete
  - Set status (Active or Inactive)

- **Predefined:**
  - Duplicate
  - Edit (which is really creating a custom version of the predefined file)
  - Set status

Navigation

This table describes where to go to customize help.

<table>
<thead>
<tr>
<th>Help Customization Task</th>
<th>Navigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create or edit help for a specific help window.</td>
<td>Click the Manage Custom Help 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 Edit link.</td>
</tr>
</tbody>
</table>
Perform any help file customization task, including managing a set of help (such as all help for a product)

Go to Applications Help and click your user name in the global area to select Manage Custom Help.

Make a copy of all custom help for testing, migration, or other purposes.

Create a configuration package then use the export and import feature 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 Custom Help: Explained

You have many options to determine the content of custom help, for example by uploading a 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 custom 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. When you read help and 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 Custom Help 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), or dashboard regions
- In the Setup and Maintenance work area, next to names of task lists or tasks in tables

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.

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.

Customizing Help

Watch: This video tutorial shows you how to add and edit help in help windows.

Customizing Help in Help Windows: Procedure

When users click help icons on a page, they open help windows that have informational text, links to help files, or both. Use the Manage Custom Help dialog box to edit those help files, create new files to appear in the help window, change the order of links in the window, or add links to existing custom help. To restrict access to any of those help files, use the Manage Custom Help page in Applications Help to assign a help security group to the help file.

Tip: You can also use:

- The Customer 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 a help window
Opening the Manage Custom Help Dialog Box

Use the Manage Custom Help dialog box to create and edit help files, or to reorder links:

1. Click the help icon to open the help window you want to customize. If you don’t see any help icons, click Show Help in the global area.
2. Click Manage Custom Help.

Editing Help Files

To edit help files in the help window:

1. Open the Manage Custom Help 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 custom version of the predefined file.

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.

Creating Help Files

To add a custom help file that appears in the help window (and also Applications Help):

1. Open the Manage Custom Help dialog box for the help window.
2. Click Create.
3. Select a source and enter 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, keywords, and description for your file.
6. Optionally select a language, country, or role, which correspond to filters in Applications Help.
7. Save your work.

Adding Links to Help Windows

To add existing custom help files as links in any help window:

1. In Applications Help, click your user name in the global area and select Manage Custom Help.
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 existing custom help file, select it, and click Apply.
6. Repeat the previous step for all the help files you want to add.
7. Click Done.
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 Custom Help page in Applications Help, not the Manage Custom Help 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 area and select Manage Custom Help.
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 above) to narrow down the list of available page or section values.

   ![Note:](image)
   
   For any help window for task lists or tasks in the Setup and Maintenance work area, select this hierarchy:
   - Hierarchy: Task
   - Level 1: Functional Setup

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 Region in Applications Help: Procedure

In Applications Help, users can use the tabs right below the global area to browse help by product family. Every product family tab has a Getting Started region, 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 custom help to the Getting Started region for any product family.

Adding Help to the Region for a Product Family

Follow these steps:

1. Click your user name in the global area in Applications Help, and select Manage Custom Help.
2. Create or edit the help file you want to add to the Getting Started region.
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 that you want the file to appear under. Optionally select a product in the Level 2 column.
7. Save your work.

Links in Custom Help: Points to Consider

When you create or edit custom 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 custom 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.

This figure provides an example of related links code.

OfaRelatedTopics(CREATE_AUTOMATIC_POSTING_CRITERIA_S_0000, JOURNAL_ENTRIES_HOW THEY RE RECORDER_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 right title IDs. To find title IDs, search for the help files on the Manage Custom Help 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
<span class="HP_topic-link_bridgeDocument-linkToSTDoc_"><?ofa linkToSTDoc(WCSUG4636) ?></span>.
```

**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.

This figure shows an example of code for a glossary term link.

```html
OfaGlossaryTerm("accounting period", ACCOUNTING_PERIOD_0001)
```

In this example, **accounting period** is the link text, or glossary term, and **ACCOUNTING_PERIOD_0001** 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 Custom Help 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.

Customizing PDF Guides: Worked Example

This example demonstrates how to customize 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 customized 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 customized 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 customized guide?</td>
<td>No, same as the original guide</td>
</tr>
<tr>
<td>Do you want to tag the customized 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. Create a custom version of the original help file, with your new PDF as the help content.

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.
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 customized 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 enter the following values.
<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 Custom User Productivity Kit Content to Help: Worked Example

This example demonstrates how to add a custom 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 one next to the Overview page title in the Scheduled Processes work area</td>
</tr>
</tbody>
</table>

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 custom 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, click and drag the Document ID column header and drop it after the Name ID 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 the custom 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 under the Deployment check box group.
4. In the Player section, select the Include HTML Web Site check box. Your custom 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 Custom Help File for the Topic

1. Open the Scheduled Processes work area.
2. Open the help window for the Overview page, and click Manage Custom Help.
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>Source</td>
<td>Oracle User Productivity Kit</td>
</tr>
<tr>
<td>File Location</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>Document ID</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>Help Type</td>
<td>Video</td>
</tr>
<tr>
<td>Topic Title</td>
<td>The name of the User Productivity Kit topic.</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 area 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 custom version.
- In help files, existing links to the glossary term automatically point to your custom version.

**Note:** If you later inactivate your custom glossary term, make sure to activate the original term so that the links still work.

FAQs for Help File Customization

What happens when I edit predefined help?

You're actually creating a new custom help file based on the predefined file. The custom 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 custom help?

Create a custom 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 Custom Help page.

1. In Applications Help, click your user name in the global area and select Manage Custom Help.
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 choice in the Setup and Maintenance work area.
3. Save your work.

Related Topics
- Creating Help Security Groups: Worked Example
- Setting Up Help Customization: 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 custom help in their search results?
If you customized 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 custom help after an upgrade for Applications Help?
Nothing happens to your custom help files. Upgrades affect only predefined help files, active or inactive.
Take a look at any inactive file that’s updated to see if you want to:
- Activate the updated version.
- Make similar edits to the custom 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 Customization section. Use an image that’s white along the entire left border, like you see in the default image.

**Note**: If you don’t see the Help Site Customization section on the Set Help Options page, check with your implementor about selecting the Help Customization feature choice.

Customization of Getting Started Pages
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 customize the predefined pages or add your own, so that users get content specific to your organization.

Access to Customization

Open the Getting Started work area, and, if you have multiple sets of pages, select the set to customize. You then see the Edit Getting Started link if you have the appropriate roles.

Key Customization 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 custom 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 Customization
Customizing Help That Appears on the Page: Highlights

You can customize help that you see on the page, for example hints for check boxes 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 on hover over buttons, links, icons, or tab titles. Open the properties of the UI element to define the help text in the shortDesc field.

- 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.

- Edit, create, or delete most types of embedded help using design time tools (not available in Oracle Cloud implementations). Refer to the Customizing or Adding Static Instructions, In-Field Notes, and Terminology Definitions section.

See: Oracle Fusion Applications Extensibility Guide for Developers

Related Topics

- Customizing Simplified Pages Using Page Composer: Procedure

- Page Component Properties: Explained

- Bulk Text Customizations: Explained
16 Activities and Sales Orders Extensibility

Overview

Learn about the Activity and Sales Order objects, and how to extend those objects using Application Composer.

Extending Simplified Pages for Activities: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for activities. Use Application Composer to customize these 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, available in the Sales application.

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 customize 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 below subtabs are available from the Edit Task, Edit Appointment, and Edit Call Report pages. Use the below subtabs to configure which standard and custom fields display at run time.

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.

- **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 in the Common application, edit the default creation page layout or duplicate it to create a new layout.

The Notes list page is not extensible.
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, available in the Common application, to customize 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.

Customize 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
• Customizing Oracle Sales Cloud Simplified Pages: Explained
17 Account, Contact, and Household Extensibility

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 customize Customer Center pages using Page Composer.

Extending Simplified Pages for Accounts: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for accounts.

Use Application Composer to customize these pages:

- Accounts landing page
- Create Account page
- Edit Account page (see discussion of subtabs below)

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 accounts, use the Account object in Application Composer, available in the Common application.

Accounts Landing Page

You can customize 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 customize 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 customize 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 customize 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 run time 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 customize most of these subtabs, as well as hide or show subtabs. You can also add custom subtabs to display records from child or related objects, for example.

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 back to the desired details page layout for the Account object to add those fields to the right subtabs.

To customize 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 below subtabs are available from the Edit Account page. Use the below subtabs to configure which standard and custom fields display at run time.

The changes you make to these subtabs are unique to the Edit Account page. For example, the changes you make to the Opportunities subtab on the Edit Account page are not reflected on the Opportunities subtab on the Edit Contact page.

- 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 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 in the Common application, 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 in the Sales application, scroll down to the bottom of the page and click **Edit Embedded Summary Table**.
  - Conversations
    - Not extensible from this page.
  - White Space Analysis
    - Sales reps 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**
- Customizing Oracle Sales Cloud Simplified Pages: Explained

**Extending Simplified Pages for Contacts: Explained**

Using Application Composer, you can change many items that appear on the simplified set of pages for contacts.

Use Application Composer to customize these pages:

- Contacts landing page:
  - Contacts cards view
  - Contacts listing
- Create Contact page
- Edit Contact page (see discussion of subtabs below)
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 contacts, use the Contact object in Application Composer, available in the Common application.

### 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 customize both pages.

The Contact cards view displays a list of all your users’ contacts, but represented visually as cards. You can customize the fields that appear in each card. For example, you can hide standard contact fields or show custom contact fields. To customize 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 the Configure Card View region, select the fields that you want to display or hide in your contact cards.
5. Optionally click 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 customize the columns that appear in the table. For example, you can hide standard contact fields or show custom contact fields. To customize 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 customize 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 customize 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 run time 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 customize most of these subtabs. You can also add custom subtabs to display records from child or related objects, for example.

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 back to the desired details page layout for the Contact object to add those fields to the right subtabs.

To customize 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 below subtabs are available from the Edit Contact page. Use the below subtabs to configure which standard and custom fields display at run time.

   The changes you make to these subtabs are unique to the Edit Contact page. For example, the changes you make to the Opportunities subtab on the Edit Contact page are not reflected on the Opportunities subtab on the Edit Account page.

   - 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
     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.
   - Activities
   - Notes list page is not extensible.
Use the Activity object in Application Composer to extend this subtab. On the Simplified Pages tab for the Activity object in the Sales application, scroll down to the bottom of the page and click **Edit Embedded Summary Table**.

- **Conversations**
  - Not extensible from this page.

**Related Topics**
- Customizing Oracle Sales Cloud Simplified Pages: Explained

## Extending Simplified Pages for Households: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for households. Use Application Composer to customize these pages:

- Households landing page
- Create Household page
- Edit Household page (see discussion of subtabs below)

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, available in the Common application.

### Households Landing Page

You can customize 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 customize 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 customize 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 customize 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 run time 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 customize most of these subtabs. You can also add custom subtabs to display records from child or related objects, for example.

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 back to the desired details page layout for the Household object to add those fields to the right subtabs.

To customize 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 below subtabs are available from the Edit Household page. Use the below subtabs to configure which standard and custom fields display at run time.

The changes you make to these subtabs are unique to the Edit Household page. For example, the changes you make to the Opportunities subtab on the Edit Household page are not reflected on the Opportunities subtab on the Edit Contact page.

- 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 in the Common application, 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 in the Sales application, scroll down to the bottom of the page and click **Edit Embedded Summary Table**.

- **Conversations**
  Not extensible from this page.

**Related Topics**
- Customizing 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 customize 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 customize 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 custom relationship fields. You can also reorder fields and change field labels. To customize 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 custom fields into field groups.
Extending Simplified Pages for Assets: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for assets.

Use Application Composer to customize these pages:

- Assets 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 simplified details pages for these objects:

- Accounts
- Contacts
- Households

**Note:** Use the Asset object in Application Composer, available in the Sales application, to create custom asset fields. Note that you cannot add custom buttons and actions to any simplified page for assets.

Assets List Page

You can customize 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 customize 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 customize 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 customize 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 run time usability for your end users, optionally group custom fields into field groups.

**Edit Asset Page**

You can customize 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 customize 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 run time usability for your end users, optionally group custom fields into field groups.

**Related Topics**

- Customizing Oracle Sales Cloud Simplified Pages: Explained

**Extending Simplified Pages for Assets: Worked Example**

This example demonstrates how you can customize 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. In Application Composer, select the Sales application on the main Overview page.
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 customize the simplified pages that are available for the selected object. You can show or hide fields, rearrange fields, and add custom 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 customize 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. In the Create Asset page that appears, notice that the field **Asset Number** that you added using Application Composer now appears at the bottom. This figure shows the **Create Asset** page with the field **Asset Number** that we added as part of extending the Asset object.

Customizing the Overview Subtab: Explained

You can customize 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 customize 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 Boxes]

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**

<table>
<thead>
<tr>
<th>Actions</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<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>

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 shuttles to control which summary boxes are displayed within this summary box group at run time. Within the Displayed Summary Boxes shuttle, use the up or down arrows to control the order of the summary boxes within the group.
   - If a summary box is listed under the Available Summary Boxes shuttle 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:

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. Scroll 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.

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 Multiple Addresses: Explained

Accounts, contacts, households, and partners automatically display a primary address region. However, you can change this to instead display a table that lists multiple addresses.
Where Does Address Information Display?
Address details for accounts, contacts, households, and partners display on their respective Profile subtabs.

For example, in the screenshot below, you can see 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 change the Primary Address region to instead display the Multiple Address region.

Enabling Multiple Addresses - Points to Consider
Before you enable multiple addresses, you must consider the following:

- Ensure that only one address region, Primary Address or Multiple Addresses, is shown on the Details pages layout.
- 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.

Enabling Multiple Addresses
To display a list of multiple addresses in a table:

1. In Application Composer, navigate to the Pages node > Simplified Pages tab for either the account, contact, household, or partner object.
2. In the Details Page Layouts region, edit the desired custom layout.
If none exists, then duplicate the standard layout and edit the resulting custom layout.

3. Navigate to the Profile tab.

In the Primary Address region and Multiple Address region, you will see either a Show or Hide link.

   a. Click **Hide** to hide the region at run time.
   b. Click **Show** to show the region at run time.

For example, to enable multiple addresses, click **Hide** in the Primary Address region, and click **Show** in the Multiple Address region.
Adding Join Fields to the Sales Account Object: Worked Example

In this example scenario, you will add the Group Type field from the Household object to the Sales Account object. Before you can do this, you must first navigate to the join that exists between the Sales Account object and the Household object, and register the Group Type field as a join field.

_tip_ The Sales Account object is available for customization 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, available in the Common application.

Adding a Join Field to the Sales Account Object

Before you can add the field to the Sales Account summary table, you must register that field as a join field, even if you create a custom field.

1. In Application Composer, select the Sales application, navigate to the Sales Account object, and then click the Joins node.
2. On the Joins page, click the join name SalesAccountToGroupJoin. (The Household object was previously known as the Group object.)
   The Join Fields page opens.
3. On the Join Fields page, click Create.
   The Create Join Fields page opens.
4. On the Create Join Field page, create a join field called Group Type.
   After you click Save and Close, you return to the Join Fields page.

Adding the Join Field to the Customer Overview Page

After you create a join field, you can add it to the Customer Overview page so your users can see the field at run time.

1. Navigate to the Sales Account object, and then click the Pages node.
2. Click the Edit Summary Table link. In the Available Fields list, select Group Type - Join Field and add it to the Selected Fields list.

_tip_ You can adjust the placement of the new column right or left in the table by moving the join field up or down in the Selected Fields list.
3. Click **Save and Close**.
4. From the Navigator menu, click **Customers**.

The new Group Type column now appears in the Sales Accounts summary table on the Customer Overview page.
Adding the Join Field to the Local Search Region

Next, add your new join field as a column in the local search region of the Customer Overview page, so your users can see the field at run time.

1. Navigate to the Sales Account object, and then click the Pages node.
2. Click the Edit Local Search link.
3. In the Available Fields list, select your new join field and move it to the Selected Fields list.

The join field appears on the Sales Account user interface.

Tip: You can adjust the placement of the new column right or left in the table by moving the join field up or down in the Selected Fields list.
The following figure shows the GroupType join field added to the Customer Overview page.

Oracle Sales Cloud Common Objects for Desktop Page Configurability: Explained

You can customize a variety of pages and regions throughout Oracle Sales Cloud using Application Composer. This topic provides an overview of each standard object in the Common application container. (This topic does not address the Note common component object also listed within the Common application container.)

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.

**Note:** You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

Objects are attached to a specific application container, such as Sales. Common objects, however, are unique in that the customizations you make to these objects can be displayed in a variety of applications, and in a variety of pages and regions, at run time.

The Common standard objects that you can extend are:

- Account
- Address
- Contact
- Customer Contact Profile
- Household
- Relationship
- Resource

**Account**

The Account object stores all intrinsic information for all accounts associated with the deploying company. Examples of accounts are customers (including sales prospects and external legal entities), competitors, and partners. To specifically extend customer or partner objects, use the Sales Account object (under the Sales application container) or the Partner
object (under the Sales application container). (The Account object is a generic profile that is applicable to all types of organizations in the system. The Sales Account object is applicable only to those organizations that are sales accounts.) Since Prospect and Legal Entity customers don’t have separate profiles, you can extend these customers only by using the Account object.

**Note:** The Sales Account object is available for customization 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, available in the Common application.

The Account object is a date effective entity, which means that every change or update that happens to any attribute of the object causes a new record to be created in the object with effective start and end dates. This maintains the profile history information for the object.

Any extensions to this object are available for display in various pages within Oracle Sales Cloud. Extend this object only when your customization is required for all accounts associated with the deploying company.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Account object.

This table indicates, for the Account object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Organization Quick Creation Region</td>
<td>• Account Information region within Customer Center</td>
</tr>
<tr>
<td>Edit Organization Details Region</td>
<td>• Customer Details region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>• Competitor Summary region within Sales</td>
</tr>
<tr>
<td>Edit Organization Read-Only Region</td>
<td>• Customer Basic Information region within Customer Center</td>
</tr>
</tbody>
</table>

**Address**

The Address object is an intersection object that stores a reference to a physical location/address, as well as a reference to an account, contact, or household. Any extensions to this object are available for display in customer, contact, partner, and resource address user interfaces within Oracle Sales Cloud.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Address object.

This table indicates, for the Address object, which Application Composer configuration page lets you populate which page or region with your application customizations.
Oracle Sales Cloud has provided multiple configuration capabilities for the Address object. Accordingly, do not use the Application Composer for these types of customizations:

- To define country-specific address formats, different labels for address elements, set address elements as required, and so on, use the Manage Address Formats task in the Setup and Maintenance work area.
- To enable the list of values and validation for each country, use the Manage Geographies task in the Setup and Maintenance work area.

Contact

The Contact object stores all intrinsic information for all persons associated with the deploying company. Examples of contacts are customers, prospects, internal employee resources, and partner members. To extend the customer contact relationship, use the Contact object. To extend internal resources, use the Resource object. To extend person customers (also called consumers), use the Sales Account object (under the Sales application container). (The Contact object is a generic profile that is applicable to all types of persons in the system. The Sales Account object is applicable only to those persons who are sales accounts.)

\[\textbf{Note:}\] The Sales Account object is available for customization 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, available in the Common application.

The Contact object is a date effective entity, which means that every change or update that happens to any attribute of the object causes a new record to be created in the object with effective start and end dates. This maintains the profile history information for the object.

Any extensions to this object are available for display in various pages within Oracle Sales Cloud. Extend this object only when your customization is required for all contacts associated with the deploying company.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Contact object.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
</table>
| Edit Address Summary Table                | Edit Account - Addresses region in Customer Center  
|                                           | Edit Contact - Addresses region in Customer Center  
|                                           | Edit Partner Profile - Address region in Partner Center  
|                                           | Edit Partner - Partner members - Address region within Partner Center  |
| Edit Address Creation Region              | Edit Account - Create Address region in Customer Center  
|                                           | Edit Contact - Create Address region in Customer Center  
|                                           | Edit Partner Profile - Address Create region in Partner Center  
|                                           | Edit Partner - Partner Members - Address Create region within Partner Center  |
| Edit Person Quick Creation Region         | Address region within Customer Center, on the Create Contact page  |
| Edit Organization Quick Creation Region   | Account Information region within Customer Center, under the non-extensible address section on the Create Account page  |
This table indicates, for the Contact object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
</table>
| Edit Manage Contacts Summary Table       | • Search Results summary table on the Manage Contacts page  
                                         | • Custom fields are also available for selection by clicking Add Fields in the Search region on the Manage Contacts page. |
| Edit Create Contact Region               | • Basic Information region within Customer Center, on the Create Contact page |
| Edit Contact Basic Info Region           | • Contact Basic Information within Customer Center, on the Edit Contact page |
| Edit Contact Additional Details Region   | • Additional Details region within Customer Center, on the Edit Contact page |
| Edit Person Details Region               | • Personal Information region within Customer Center |

Customer Contact Profile

The Customer Contact Profile object stores all intrinsic information for all customer contact persons, in the context of the contact’s relationship with a customer. For extensions specific to a customer contact relationship, use this object. By contrast, a standalone contact in Oracle Sales Cloud is a contact not tied to any customer and cannot be extended using this object. Instead, for extensions that apply to all person parties, use the Contact object.

Any extensions to this object are available for display in various pages within Oracle Sales Cloud.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Customer Contact Profile object.

This table indicates, for the Customer Contact Profile object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
</table>
| Edit Contact Summary Table               | • Edit Account - Contacts table in Customer Center  
                                         | • Edit Partner - Members table in Partner Center |
| Edit Contact Details Region              | • Edit Account: Contact Details within Customer Center  
                                         | • Edit Partner - Partner Members - Person Details region within Partner Center |
| Edit Related Customers Summary Table     | Related Customers summary table on the Edit Contact Profile page, in Customer Center |
| Edit Related Customers Creation Region   | Create Customer Relationship dialog, on the Edit Contact Profile page, in Customer Center |
| Edit Related Customers Edit Region       | Edit Customer Relationship dialog, on the Edit Contact Profile page, in Customer Center |
Household

The Household object stores all intrinsic information for all households associated with the deploying company. In Oracle Sales Cloud, households can be customers of the deploying company. The Household object is a generic profile that is applicable to all types of households in the system. The Sales Account object is applicable only to those households that are sales accounts. The Household object is a date effective entity, which means that every change or update that happens to any attribute of the object causes a new record to be created in the object with effective start and end dates. This maintains the profile history information for the object.

The Sales Account object is available for customization 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, available in the Common application.

**Tip:** Extend this object only when your customization is required for all households associated with the deploying company.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Household object.

This table indicates, for the Household object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Customer Center Group Pages &gt; Edit Creation Page</td>
<td>Group Customer Basic Information region on the Create Household page, in Customer Center</td>
</tr>
<tr>
<td>Edit Customer Center Group Pages &gt; Edit Details Page - Details Region</td>
<td>Additional Details region on the Edit Household page, in Customer Center</td>
</tr>
</tbody>
</table>

Relationship

The Relationship object stores all intrinsic relationship information for all kinds of relationships between parties. Use this object to add attributes specific to a relationship between any two parties.

Though a customer contact is also a type of relationship, use the Customer Contact Profile object for customer contact relationship-specific extensions. Use this Relationship object to extend only non-contact types of relationships.

Relationship Profile pages are regions that are visible in Customer Center and also in the Customer Hub. Accordingly, any extensions to this object are available for display in various pages within Oracle Sales Cloud. Generic person extensions are also visible from the Oracle Sales Cloud Customer Hub service in the Person work area.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Relationship object.
This table indicates, for the Relationship object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Relationship Summary Table</td>
<td>• Relationships summary table within Customer Center</td>
</tr>
<tr>
<td>Edit Relationship Creation Region</td>
<td>• Create Relationship dialog within Customer Center</td>
</tr>
</tbody>
</table>

**Resource**

The Resource object defines users of Oracle Sales Cloud. They are either internal employee and contractor resources, or partner member resources. Use this object when specific extensions are required for persons in the system who are resources.

To display your extensions, such as custom fields, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in Application Composer from the Pages node under the Resource object.

This table indicates, for the Resource object, which Application Composer configuration page lets you populate which page or region with your application customizations.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Local Search</td>
<td>• Search region on the Manage Resources page, available from the Manage Resources task in the Setup and Maintenance work area.</td>
</tr>
<tr>
<td>Edit Summary Table</td>
<td>• Resource Details region on the Add Resource Information page, available from the Manage Resources task in the Setup and Maintenance work area.</td>
</tr>
<tr>
<td>Identify Resource Form</td>
<td>• Resource Details region on the Add Resource Information page, available from the Manage Resources task in the Setup and Maintenance work area.</td>
</tr>
<tr>
<td>Edit Summary Form</td>
<td>• Resource page, available in the Resource Directory work area or from the Manage Resources task in the Setup and Maintenance work area.</td>
</tr>
</tbody>
</table>

**Related Topics**

- Accessing View Objects in Scripts: Explained
- Global Functions: Explained
- Defining Pages: Explained
Customizing the Customers Work Area in the Desktop UI Using Page Composer: Explained

You can customize the Customers work area in the desktop UI using Page Composer.

To access Page Composer:

1. Navigate to the work area by selecting **Customers** from the Navigator menu.
2. You must perform your customization in a sandbox, so make sure that you have a sandbox active.
3. Click on the user name in the toolbar, and select **Customize Customers Pages** under the **Administration** heading.

> **Note:** To access Page Composer in Customers work area pages, you must have the Sales Administrator job role.

In the Customers work area and pages, you can perform customizations for the enabled components in the Design and the Select modes.

Customization Tasks in Design Mode

You can perform the following customization tasks in the Customers work area and pages in Design mode. You can make the customizations available for the Site and Job Role layers.

- Change the default saved search.
- Edit saved searches.
- Add or remove predefined content from the Customers Analytics subtab.
- Customize what information appears for a customer in the customer tree:
  - Hide or show nodes.
  - Change the default node.
  - Expand or collapse nodes.
  - Specify what information is displayed by default.
- Add or remove predefined content on the Customer Snapshot page.
- Customize the default values in the lists of values, for example, in the Regional Search region.

Customization Tasks in the Select Mode

In the Select mode, you can perform the following customization tasks for enabled components:

- Manage the Customers tree.
- Hide or show fields.
- Change field labels.
- Make a field required.
- Make a field read-only or updatable.
- Reorder fields in a form.
• Reorder table columns.
• Hide or show table columns.
• Set the width and minimum width for a table column in percent or pixels.
• Enable or disable column sorting.
• Edit fields within lists.

Adding Custom Reports
You can add custom reports to the Customers work area. Here is how:

1. You create the report using BI Answers.
2. Save the report to the Resource Catalog.
3. Use Page Composer to add the report to one of two pages:
   - The Customer Snapshot
   - The Analysis tab on the Overview page

\[\text{Note:}\] The Customer Snapshot provides information on a single customer, so the reports must do so too. The reports that you add to the Analysis tab on the Overview page can provide context across multiple customers.

Related Topics
• Extending Simplified Pages for Activities: Explained

Customizing Customer Information Display in the Desktop UI: Worked Example

Use this example to learn how to customize what information users see by default while editing a customer record in the desktop UI.

To specify what customer information displays by default when a user drills down on a customer record, you can edit the customer tree, shown in the following image, while you are in the Page Composer Design mode. Using Page Composer, you can provide different views for different job roles.

\[\text{Note:}\] You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.
Before you customize 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 customizations 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 customizations separate from other customizations, then create a new sandbox:
   a. Click Create.

   ![Create Sandbox](image)

   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.

Customizing Customer Center Tree Nodes

1. Select Customers from the Navigator.

   The Overview page appears.

2. Search for any customer and drill down on the record to edit it.

   The Edit Account page appears.
3. Click on the user name in the toolbar, and select **Customize Customers Pages** under the **Administration** heading.

4. In the Customize Customers Pages dialog box, specify the set of users that will be affected by this customization by choosing one of the following:
   - **Site** to make the customization available to all users
   - **External or Internal** to make the customization available to either internal (employee) users, or to external users such as partners.
   - **Job Role** to make the customization available to users with a job role you specify.

5. Click **OK**.
The page opens in the Page Composer Design mode.

6. In the Customer Center pane, select **Manage Customer Tree** from the Actions menu.

The Manage Customer Tree window appears.

Using the View menu, you can show or hide different nodes or expand them. Using the Actions menu, you can set any of the nodes to display by default whenever a user edits an account.

7. To change the default display to the customer snapshot, for example, select the **Snapshot** row in the Tree Nodes table and select **Set as Default** from the Actions menu.

8. Click **Save** after each customization.

9. Click **Save and Close** when you are done.
10. Click **Close** to exit from Page Composer.

The customer tree reflects your customizations immediately, but they will not be seen by users until you publish the sandbox.

**Changing the Label and Width of a Table Column Using Page Composer: Worked Example**

Use this example to learn how to customize table column labels and the width of table columns in desktop UI pages. This example covers changing the properties of the table of search results in the Customers Overview page in the desktop UI.

Before you customize 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 customizations 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 customizations separate from other customizations, then create a new sandbox:
   
   a. Click **Create**.

   ![Create Sandbox](image)

   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.
Changing the Label and the Width of a Column in the Table Listing Customers

1. Select Customers from the Navigator.

The Customers Overview page appears.

2. Click on the user name in the toolbar, and select Customize Customers Pages under the Administration heading.

3. In the Customize Customers Pages dialog box, specify the set of users that will be affected by this customization by choosing one of the following:
   - Site to make the customization available to all users
   - External or Internal to make the customization available to either internal (employee) users, or to external users such as partners.
4. Click **OK**.

The page opens in the Page Composer Design mode.

5. Click **Select**.

You are now ready to customize the table. In this example, you want to:

- Change the column heading label from **Sell-to Address** to **Address**.
- Change the width of the column

6. Mouse over the **Sell-to Address** column header.

The column header is automatically outlined with a blue border.

7. Click within the border.

A menu appears with two selections: **Edit Component** and **Edit Parent Component**.

8. Select **Edit Component**.
The Component Properties window appears. The numbers in the following image identify the customizations you are making.

9. To change the column header text:
   a. Click on the down arrow to the right of the **Header Text** field (identified by 1 in the previous image) and select **Select Text Resource**.
The Select Text Resource window appears.

b. Enter a value in the **Key** field. The key identifies the text label you are creating, so you can search for it and reuse the label in other pages.

c. Enter the new column label in the **Display Value** field. In this example, you enter **Address**.

d. Click **OK**.

10. Click **Close** to exit from Page Composer.

The column reflects your customizations immediately, but they will not be seen by users until you publish the sandbox.
18 Leads and Marketing Extensibility

Overview

Read this chapter to learn about extending the sales lead and marketing objects and pages in Oracle Sales Cloud. In this chapter, you will learn:

• How to customize Oracle Sales Cloud leads and marketing 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

Some topics in this chapter address how to customize the desktop version of Oracle Sales Cloud marketing pages.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

Sales Lead and Sales Campaign

Extending Simplified Pages for Leads: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for sales leads.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

Use Application Composer to customize these pages:

• Leads landing page (Landing Page Layouts)
• Create Lead page (Creation Page Layouts)
• Edit Lead Page (Details Page Layouts)
• Lead E-mail 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 custom fields (all types).

Note: To extend the simplified set of pages for leads, use the Sales Lead object in Application Composer, available in the Sales application. 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 custom fields. To customize 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 customize the Create Lead page. For example, you can hide standard fields, make a field required, add a default value, and show custom fields. To customize 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 run time usability for your end users, optionally group custom 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 customize most of these subtabs. You can also add custom subtabs, for example, to display records from child or related objects.

To add custom fields to certain subtabs available from the EditLead 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 Lead object to add those fields to the specific subtabs.

To customize 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 custom 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 custom fields display at run time.

- Summary
- Contacts

Use the Sales Lead Contact object to create custom fields as needed.
Qualification
Not extensible from this page.
Sales Team
Use the Sales Lead Resources object to create custom fields as needed.
Products
Use the Sales Lead Products object to create custom fields as needed.
Activities
Use the Activity object in Application Composer to extend this subtab.
On the Simplified Pages tab for the Activity object in the Sales application, 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 custom fields as needed, and to extend this subtab.
On the Simplified Pages tab for the Note object in the Common application, 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 custom fields as needed.
Social
Not extensible from this page.

Lead E-Mail Notifications
This region is applicable only for partner leads. From here, you can customize actionable e-mail content that is sent to users when partner leads are assigned. Use the Sales Lead object in Application Composer to extend the e-mail notifications of the partner lead UI. On the Simplified Pages tab for the Sales Lead object in the Sales application, 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
- Customizing Oracle Sales Cloud Simplified Pages: Explained

Extending Simplified Pages for Leads Using Application Composer: Worked Example
You can extend simplified pages using Application Composer. You must make your changes in a sandbox, so you can test them first.
Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

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 and select Sales from the Application list. 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:

```groovy
def retVal
if (StrategicDeal_c == 'Y')
{
    if (DealAmount >= 1000)
    {
        retVal = true
    } else
    {
        retVal = false
    }
}

if (StrategicDeal_c == null || StrategicDeal_c == 'N')
{
    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

Customizing Sales Campaign Pages: Explained

Use Application Composer to create custom fields for the sales campaign object. You can then add the custom fields for display in the Sales Campaign work area. This topic describes the extensible options for sales campaigns.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps

To customize the sales campaign work area, follow these steps:

1. Navigate to Application Composer, select Sales as the Application, expand the Standard Objects tree node, then select Sales Campaign.
2. Select the **Fields** node to create custom fields, or modify standard fields.
3. Select the **Pages** node to show or hide fields, either custom or standard.
4. Use the links on the tab to navigate to the object’s configuration pages. From there, you can customize the pages that are available for the selected object.

**Note:** For simplified pages, 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 Extensible**

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 simplified pages are extensible, 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>
</tbody>
</table>

The following list shows which regions and desktop pages are extensible, 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>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 custom fields to the Sales Campaign regions listed, first create your custom fields by selecting the Fields node under the sales campaign object. Then, select either a standard field to modify, or create a custom 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 custom fields for the sales campaign object, use Application Composer’s configuration pages to add the custom fields to the extensible 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. Select the Sales application on the main 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 customize.

Notes About Sales Campaign Extensibility

The following customization 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 customization options do not apply to sales campaigns:

- Role security
- Custom subject area.

Related Topics

- Defining Pages: Explained
- Defining Fields: Explained
- Customizing Sales Cloud Applications Across Application Boundaries: Explained
- Sales Campaigns: Overview

Customizing Sales Lead Pages: Explained

Use the Application Composer to create custom fields for the lead object, which you then add for display in the Lead work area. This topic describes the extensible options for sales lead and associated objects.
Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

This topic addresses the configurability of sales lead desktop pages only. To understand how to configure the set of simplified pages for sales lead, see the related topics on simplified pages.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

The objects that are associated with the lead management application are:

- Sales Leads
- Sales Lead Contacts (child object of Sales Lead)
- Sales Lead Products (child object of Sales Lead)
- Sales Lead Resources (child object of Sales Lead)
- Marketing Response (related object of Sales Lead)

Note: If you are customizing the Sales Leads object and your organization is using the Sales Prediction Engine (SPE) to generate leads, you must set default values for any custom required fields on the Sales Leads object. SPE does not generate values for custom required lead attributes, therefore to successfully use the SPE Lead Generation feature to create leads, make sure you supply a default value for any customized Lead attribute which is marked Required.

Extensibility: Quick Steps
To customize the sales lead work area, follow these steps:

1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand Sales Lead, Sales Lead Contacts, or Sales Lead Products, or Sales Lead Resources nodes.

   Note: If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the Fields node to create custom fields, or modify standard fields.
3. Select the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Sales Lead work area to view your changes.

Understanding Which Sales Leads Pages Are Extensible
To customize Sales Leads pages, you must know which pages and regions are extensible, and which objects to select in the Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages show additional details about an individual object record

The following table lists Sales Lead pages, regions, child objects and the related object that you can customize using Application Composer. For example, the Sales Lead Contacts tab on the Edit Sales Lead page is a child object of the Sales Lead object and is extensible. To create custom fields that you can later add to the Sales Lead Contacts tab, you must select the Sales Lead Contacts object in the Application Composer and create your custom fields.
Next, expose those custom fields on the Sales Lead page by accessing the appropriate Application Composer configuration page listed in the following section.

To make only minor user interface changes to Sales Leads pages without creating objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields
To add custom fields to the Sales Lead regions listed above, first create your custom fields by selecting the Fields node under the sales lead, or sales contact, or sales product, or sales resources object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application
To add custom fields to the Sales Lead Contacts region listed in the table above, first create your custom fields using the Application Composer. Next, use the Application Composer’s configuration pages to add those custom fields to the Sales Leads regions that you want. You access the configuration pages in the Application Composer from the Pages node under the Sales Lead object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page
2. In the object tree, select the Sales Lead object.
3. Select the Pages node.

<table>
<thead>
<tr>
<th>Sales Leads Configuration</th>
<th>Sales Leads UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Lead Qualification, Lead table</td>
</tr>
<tr>
<td>Create Lead creation page</td>
<td>Lead Qualification, Create Lead page</td>
</tr>
<tr>
<td>Registration Details creation page</td>
<td>Lead Qualification, Create Lead, Registration Details region (select Partner as the Sales Channel)</td>
</tr>
<tr>
<td>Show Details details page</td>
<td>Lead Qualification, Edit Lead, Summary region (expand Show Details to view more details)</td>
</tr>
<tr>
<td>Basic Qualification details page</td>
<td>Lead Qualification, Edit Lead (expand the Basic Qualification option to view details)</td>
</tr>
<tr>
<td>Registration Details details page</td>
<td>Lead Qualification, Edit Lead (when Sales Channel is Partner), Expand the Registration Details option to view details.</td>
</tr>
<tr>
<td>Edit Lead details page</td>
<td>Lead Qualification, Edit Lead, Summary region</td>
</tr>
<tr>
<td>Contacts lead tab component</td>
<td>Lead Qualification, Edit Lead, Contacts tab</td>
</tr>
<tr>
<td>Products lead tab component</td>
<td>Lead Qualification, Edit Lead, Products tab</td>
</tr>
<tr>
<td>Resources lead tab component</td>
<td>Lead Qualification, Edit Lead, Sales Team tab</td>
</tr>
<tr>
<td>Marketing Response lead tab component</td>
<td>Lead Qualification, Edit Lead, Responses tab</td>
</tr>
</tbody>
</table>
4. On the Pages Overview page, select the configuration page link related to the Sales Lead region that you want to customize.

**Related Topics**
- Adding a Field to a Sales Lead Page: Worked Example
- Customizing Oracle Sales Cloud Simplified Pages: Explained

**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.

> **Note:** You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

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. From the Application field, select Sales.
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:
   
   ```
   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:

   **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 select Sales from the Application list.

4. Expand Standard Objects, then Sales Lead and then click Server Scripts.

5. Click the Triggers tab, and then click the Create icon in the Object Triggers section.

6. 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:
   ```java
   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);
            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;
               }
            }
            if(!alreadyOnSalesTeam) {
               def leadResourceVO = leadResourceIterator.createRow();
               leadResourceVO.setAttribute('LeadId',LeadId);
               leadResourceVO.setAttribute('ResourceId',resourceId);
               leadResourceVO.setAttribute('PrimaryFlag','Y');
            }
         }
      }
   }
   ```

7. Click Save and Close. Verify that you see the newly created trigger in the Object Triggers section.

8. Next, click the Create icon in the Object Triggers section to create another object trigger.

9. 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)
         {  
```
setAttribute('OwnerId' , accRow?.OwnerPartyId);
def resourceId = accRow?.OwnerPartyId;
boolean alreadyOnSalesTeam = false;
leadResourceId?.reset();
for (; leadResourceId?.hasNext(); ) {
def mklLeadResourcesVORowImpl = leadResourceIdIterator.next();
Long existingResourceId = mklLeadResourcesVORowImpl.getAttribute('ResourceId');
if (resourceId.equals(existingResourceId)) {
  alreadyOnSalesTeam = true;
break;
}
if(!alreadyOnSalesTeam) {
def leadResourceVO = leadResourceIdIterator.createRow();
leadResourceVO.setAttribute('LeadId',LeadId);
leadResourceVO.setAttribute('ResourceId',resourceId);
leadResourceVO.setAttribute('PrimaryFlag','Y');
}
}

10. Click **Save and Close**. Verify that you see the newly created trigger in the **Object Triggers** section.

11. Thoroughly test the following scenarios in your sandbox environment:
   o When a lead is created with an associated account and the assignment process is run
   o When a lead is created without an account, but the account is added later and the assignment process is run

12. Publish the sandbox after successfully verifying your tests.

**Related Topics**
- Sandboxes: Explained

Customizing 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 customization 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:* To perform this customization, you must 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 select **Sales** from the **Application** list.
2. Expand **Standard Objects**, then **Sales Lead** and then click **Fields**.
3. Search for the **Primary Contact** field of type **Dynamic Choice List**.
4. From the **Edit Standard Field: Primary Contact** page, under the **Constraints** section, select the **Required** check box.
5. Click Save and Close.

Hide the Name Field

Hide the Name field on the leads UI as follows:

1. Navigate to Application Composer and select Sales from the Application list.
2. Expand Standard Objects, then Sales Lead and then click Pages.
3. Navigate to Sales Lead > Pages > Creation Page Layout > Create a Duplicate Layout page.
4. From the Creation Page Layout section, select Duplicate Layout from the Actions menu.
5. In the Duplicate Layout dialog, enter Default Custom Layout in the New Layout Name field.
6. Click Save and Edit.
7. Click the Edit icon to display the Configure Detail Form.
8. Move Name from Selected Fields list to Available Fields list.
9. Click Save and Close, then click Done.
10. 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 select Sales from the Application list.
2. Expand Standard Objects, then Sales Lead and then click Server Scripts.
3. Click the Triggers tab, and then click the Add a new Trigger icon in the Object Triggers section.
4. In the Trigger field, select the Before Insert in Database trigger type.
5. In the Trigger Name field, enter a name, without spaces, such as UpdatLeadOwnerOnCreate.
6. Copy and paste the following content of the script to the Trigger Definition section:

   ```java
   if(PrimaryContactName == null)
   {
       throw new oracle.jbo.ValidationException('Primary Contact is a required field. Please select a Primary Contact.')
   }
   else
   {
       setAttribute('Name',PrimaryContactName)
   }
   
   Click Save and Close.
   
   Verify that you see the newly created trigger in the Object Triggers section.

8. Next, click the Add a new Trigger icon in the Object Triggers section to create another object trigger.
9. In the Trigger field, select the Before Update in Database trigger type.
10. In the Trigger Name field, enter a name such as UpdatLeadOwnerOnUpdate.
11. Copy and paste the following content of the script to the Trigger Definition section:

   ```java
   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)
   }
   
   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:
14. Publish the sandbox after successfully verifying your tests.

Marketing

Customizing Marketing Campaign Pages: Explained

Use Application Composer to create custom fields for the Marketing Campaign object. You can then add custom fields for display in the Campaigns work area. This topic describes the extensible options for marketing multistage campaigns.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps

To customize the Campaigns work area, follow these steps:

1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand the Marketing Campaign node.

Note: If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the Fields node to create custom fields, or modify standard fields.

3. Select the Pages node to show or hide fields, either custom or standard.

4. Test your work by navigating to the Campaigns work area to view your changes.

Understanding Which Marketing Campaign Pages Are Extensible

To customize marketing campaign pages, you must know which pages and regions are extensible, and which objects to select in the Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record

The following table lists marketing multistage campaign pages and regions, and the related objects that you can customize using Application Composer. To create custom fields that you can later add to any of these tabs, you must select the Marketing Campaign object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Campaign Configuration</th>
<th>Multistage Campaign UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview</td>
<td>Campaigns, Overview, Marketing Campaigns tab, marketing campaigns summary region</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Campaigns, Create Multistage Campaign region</td>
</tr>
</tbody>
</table>
Next, expose those custom fields on the Campaigns work area by accessing the appropriate Application Composer configuration page, listed in the following section.

Note: To make only minor user UI changes to Marketing Campaign pages without creating objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields

To add custom fields to the marketing campaign regions, first create your custom fields by selecting the Fields node under the Marketing Campaign object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application

Use the Application Composer’s configuration pages to add custom fields to the Campaigns regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Campaign object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing Campaign object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Campaigns region that you want to customize.

Customizing Marketing Campaign Stage Pages: Explained

Use Application Composer to create custom fields for the Marketing Campaign Stages object. You can then add custom fields for display in the Campaigns work area. This topic describes the extensible options for marketing campaign stages.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps

To customize the Campaign Stages work area, follow these steps:

1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand the Marketing Campaign Stages node.

   Note: If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the Fields node to create custom fields, or modify standard fields.
3. Select the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Campaign Stages work area to view your changes.

Understanding Which Marketing Campaign Stage Pages Are Extensible

To customize pages for marketing campaign stages, you must know which pages and regions are extensible, and which objects to select in the Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record

The following table lists marketing campaign stage pages that you can access in Application Composer. To create custom fields that you can later add to any of these regions, you must select the Marketing Campaign Stages object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Campaign Stages Configuration</th>
<th>Marketing Campaign Stages UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Creation Page</td>
<td>Campaigns, Create Stage, Overview region</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Campaigns, Edit Stage details region</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing Campaign Stages page by accessing the appropriate Application Composer configuration page listed in the following section.

🚀 **Note:** To make minor UI changes to Campaign Stages pages without creating objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields

To add custom fields to the marketing campaign stages regions, first create your custom fields by selecting the Fields node under the Marketing Campaign Stages object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application

Use the Application Composer's configuration pages to add custom fields to the campaign stage regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Campaign Stages object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing Campaign Stages object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Campaign Stages region that you want to customize.
Customizing Marketing Campaign Member Pages: Explained

Use Application Composer to create custom fields for the Marketing Campaign Member object. You can then add the custom fields for display in the Campaigns work area. This topic describes the extensible options for marketing campaign members.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps

To customize fields in the Campaign Member work area, follow these steps:

1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand the Marketing Campaign Member node.

   Note: If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the Fields node to create custom fields, or modify standard fields.
3. Select the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Campaign Member work area to view your changes.

Understanding Which Marketing Campaign Member Pages Are Extensible

To customize marketing campaign member pages, you must know which pages and regions are extensible, and which objects to select in the Application Composer. Pages represent the ways in which object information is displayed in the UI.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record

The following table lists marketing campaign member pages and regions that you can customize using Application Composer. To create custom fields that you can later add to any of these pages, you must select the Marketing Campaign Member object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Campaign Member Configuration</th>
<th>Marketing Campaign Member UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Campaigns, Results tab, Execution Details, Campaign Members region</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing Campaign Member page by accessing the appropriate Application Composer configuration page listed in the following section.

Note: To make only minor UI changes to Campaign Member pages without creating objects or fields, use Page Composer instead of the Application Composer.
Adding or Modifying Fields
To add custom fields to the marketing campaign member regions, first create your custom fields by selecting the Fields node under the Marketing Campaign Member object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application
Use the Application Composer's configuration pages to add custom fields to the Campaign Member regions. Access the configuration pages from the Pages node under the Marketing Campaign Member object.

To access the Pages Overview page:
1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing Campaign Member object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Campaign Member region that you want to customize.

Customizing Marketing List Pages: Explained
Use Application Composer to create custom fields for the Marketing List object. You can then add the custom fields for display in the Audience work area. Marketing List does not have any child objects. This topic describes the extensible options for marketing lists.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps
To customize the audience work area, follow these steps:
1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand the Marketing List node.

Note: If you can't see the choice list of applications, make sure the regional area of the page is exposed.
2. Select the Fields node to create custom fields, or modify standard fields.
3. Select the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Audience work area to view your changes.

Understanding Which Marketing List Pages Are Extensible
To customize marketing list pages, you must know which pages and regions are extensible, and which objects to select in the Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:
- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record
The following table lists Marketing List pages and regions that you can access in the Application Composer to customize those pages. To create custom fields that you can later add to any of these pages, you must select the Marketing List object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing List Configuration</th>
<th>Audience UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Audience, Overview, Lists tab, List table details</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Audience, Overview, Lists tab, Create/Edit List, Summary region</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing List page by accessing the appropriate Application Composer configuration page listed in the following section.

Note: To make only minor UI changes to Marketing List pages without creating objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields

To add custom fields to the Audience regions, first create your custom fields by selecting the Fields node under the Marketing List object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application

Use the Application Composer's configuration pages to add custom fields to the Audience work area. Access the configuration pages in the Application Composer from the Pages node under the Marketing List object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing List object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Audience work area that you want to customize.

Customizing Marketing Treatment Pages: Explained

This topic describes the extensible options for marketing treatments. Use Application Composer to create custom fields for the Marketing Treatment object. You can then add the custom fields for display in the Treatment work area.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

Extensibility: Quick Steps

To customize the Treatment work area, follow these steps:

1. Navigate to the Application Composer, select Sales as the Application, expand the Standard Objects tree node, then expand the Marketing Treatment node.
Note: If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the Fields node to create custom fields, or modify standard fields.
3. Select the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Treatment work area to view your changes.

Understanding Which Marketing Treatment Pages Are Extensible

To customize marketing treatment pages, you must know which pages and regions are extensible, and which objects to select in the Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record

The following table lists marketing treatment pages and regions, and the related objects that you can access in the Application Composer to customize those pages. To create custom fields that you can later add to these pages, you must select the Marketing Treatment object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Treatment Configuration</th>
<th>Marketing Treatment UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Treatments, Overview, Treatment list table</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Treatments, Overview, Create Treatment Treatment Information region</td>
</tr>
<tr>
<td></td>
<td>Treatments, Overview, Edit Treatment Treatment Information region</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing Treatment pages by accessing the appropriate Application Composer configuration page listed in the following section.

Note: To make only minor UI changes to Treatment pages without creating objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields

To add custom fields to the marketing treatment regions listed above, first create your custom fields by selecting the Fields node under the Marketing Treatment object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application

Use the Application Composer’s configuration pages to add custom fields to the Treatment regions. Access the configuration pages from the Pages node under the Marketing Treatment object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing Treatment object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Treatment region that you want to customize.

Customizing Marketing Response Pages: Explained

Use Application Composer to create custom fields for the Marketing Response object. You can then add the custom fields for display in the Response Processing work area. Marketing Response does not have any child objects. This topic describes the extensible options for marketing responses.

> **Note:** You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

To access Application Composer, select **Application Composer** from the **Navigator** menu, under the **Tools** category.

**Extensibility: Quick Steps**

To customize the response processing work area, follow these steps:

1. Navigate to **Application Composer**, select **Sales** as the Application, expand the **Standard Objects** tree node, then extend the **Marketing Response** node.

   > **Note:** If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Select the **Fields** node to create custom fields, or modify standard fields.
3. Select the **Pages** node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Response Processing work area to view your changes.

Understanding Which Marketing Response Pages Are Extensible

To customize marketing Response pages, you must know which pages and regions are extensible, and which objects to select in Application Composer to customize those pages. Pages represent the ways in which object information can be displayed.

For example, use:

- Landing pages to show lists of object records
- Creation pages to create object records
- Detail pages to show additional details about an individual object record

The following table lists marketing response pages and regions that you can customize using Application Composer. To create custom fields that you can later add to any of these tabs, you must select the Marketing Response object in Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Response Configuration</th>
<th>Response Processing UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edit Summary Table</strong> overview page</td>
<td><strong>Manage Responses, Responses</strong>, Responses table details</td>
</tr>
<tr>
<td><strong>Edit Summary Form</strong> details page</td>
<td><strong>Manage Responses, Create Response, Summary</strong> region</td>
</tr>
<tr>
<td></td>
<td><strong>Manage Responses, Edit Response, Summary</strong> region</td>
</tr>
</tbody>
</table>
Next, expose those custom fields on the Manage Response page by accessing the appropriate Application Composer configuration page listed in the following section.

Note: To make only minor user UI changes to Manage Response pages without creating objects or fields, use Page Composer instead of Application Composer.

Adding or Modifying Fields
To add custom fields to the Marketing Response regions listed above, first create your custom fields by selecting the Fields node under the marketing response object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Run Time Application
Use Application Composer’s configuration pages to add custom fields to the Manage Response regions. You access the configuration pages in Application Composer from the Pages node under the Marketing Response object.

To access the Pages Overview page:
1. Select the Sales application on the main Overview page.
2. In the object tree, select the Marketing Response object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page link related to the Manage Response region that you want to customize.

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. 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).

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 Tools category.
3. In the Application field, select Sales, 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>
</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>
</tbody>
</table>
| Associated View Object and Associated Attribute | If you selected Join in the **Query Type** field, then use these two fields to specify the following:  
  - The view object to which this object is joined  
  - The attribute used as the basis of the join |

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>
<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 **Partner Lead to Opportunity Mapping** profile option to specify the mapping file name created when a partner lead is converted 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 create leads and opportunities from responses. You can also 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 **Tools** category.
3. In the **Application** field select Sales, and under **Advanced Setup**, select **Copy Maps**. The **Copy Maps** window appears.
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:
   ```groovy
   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.

**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 custom 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>General Lead Attribute</td>
<td>Opportunity Attribute</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------</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></td>
<td>This attribute value is the Lead Name and the current date and time, which generates a unique opportunity name.</td>
</tr>
<tr>
<td>Owner</td>
<td>Owner of opportunity</td>
</tr>
<tr>
<td></td>
<td>Indicates the user who starts the conversion.</td>
</tr>
<tr>
<td>Primary Contact</td>
<td>Key Contact</td>
</tr>
<tr>
<td>Source Code</td>
<td>Source</td>
</tr>
<tr>
<td></td>
<td>If a campaign or any other source creates a lead in which source code is used, then this attribute value is mapped.</td>
</tr>
</tbody>
</table>

### 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>
</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>
<tr>
<td>Contact Role</td>
<td>Contact Role</td>
</tr>
<tr>
<td>Primary</td>
<td>Primary</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>
Profile Options for Mapping File Names

As an administrator, you can specify your own custom mapping file name to use for mapping objects and attributes during the lead-to-opportunity conversion process. For example, you can create custom 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 custom 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.

<table>
<thead>
<tr>
<th>Lead Product Attribute</th>
<th>Opportunity Revenue Line Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Measure</td>
<td>Unit of Measure</td>
</tr>
<tr>
<td>Unit Price</td>
<td>Unit Price</td>
</tr>
</tbody>
</table>
Overview

This document explains how you use Application Composer to customize the Outlook pages supported by certain standard objects.

In this chapter, you will learn:

- Which Outlook pages are extensible, and how you can customize 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 custom objects
- What the different Outlook setup options are

Customizing 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 customizations 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.

Customizing Outlook Pages Using Application Composer: Procedure

To customize Outlook pages using Application Composer:

1. Sign in to Oracle Sales Cloud applications as a Sales Administrator.
2. Ensure you are working in a sandbox.
3. Click the Navigator icon.
4. Click Customization, and then click Application Composer.

The Application Composer page appears.
5. Select the application from the **Application** list. For example, select **Sales**.

![Application Composer](image)

6. In the object tree, select a standard object that offers a set of Outlook pages, such as **Opportunity**.

7. Select the **Pages** node.

![Application Composer](image)

8. Select the **Outlook Pages** tab.
The Outlook Pages links appear.

9. Use the options on Outlook Pages, such as Form Regions, to navigate to the object’s configuration pages, where you can customize the Outlook pages that are available for the selected object.
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** under **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**.

![Edit All Opportunities](image)

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.

---

**Customizing 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 customization 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:

- 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. 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: Fields that are displayed out-of-the-box cannot be moved 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. Click the Edit Layout icon. 
   The Edit Opportunity Form page appears.

   Note: If you want to create a common form layout for all users, use the standard layout. To create form layouts for specific user roles, see the topic on creating role-based form layout.

3. Click Add New Section.
   The Add New Form Section page appears.

4. 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.
5. Select the number of columns you want in the new section from the **Selected Layouts** options. For example, select **Two column**.

6. Click **Save and Close**.

The new section is added to the Edit Opportunity Form page.

7. Click the **Edit** icon on the first column.

The Edit Opportunity Details Column 1 page appears.

8. 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.
9. Enter the **Region Name**. For example, enter Opportunity Details.
10. 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**.
11. Click **Save and Close**.

The Edit Opportunity Form page appears with the new form section added.

12. Similarly, select and add fields **Country**, **City**, and **Created By** for column 2 of the new region.
13. Synchronize all changes in Outlook. See the Synchronizing and Validating Changes: Explained topic for details.
14. 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 views 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.

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 Role-based Form Layout: Procedure

You can create forms to be customized 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.

   ![Duplicate Layout dialog box](image)

   **Duplicate Layout**
   
   Enter the new layout name and select the existing page layout to duplicate.

   - **New Layout Name**
   - **Source Layout**: Standard layout

   **Save and Edit**  **Save and Close**  **Cancel**

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.

   ![Select: Roles dialog box](image)

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 customize 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 out-of-the-box 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.
The following figure shows an example of the parent and child objects:

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. Complete the fields in the Direct Filter Criteria region, as follows:
Field | Value  
--- | ---  
Field Name | Close Date  
Operator | Between  
Value | Literal Value  
| Current date and the date before six months.  
| For example: 10/16/14 and 4/16/14.

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.

![Configure Synchronization Filters](image)

Similarly, you can apply other filters on the server-side for any selected object.
Creating 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 customized or predefined. Customized 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 following event script types:

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Create</td>
<td>Called when an item is about to be created. Can be canceled if the context is not correct.</td>
</tr>
<tr>
<td>After Create</td>
<td>Called after an item is created, to perform custom 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.</td>
</tr>
<tr>
<td>Before Update</td>
<td>Called when an item is about to be updated. Can be used to update data before saving to enable custom logic on the form. Can be canceled if any checks fail.</td>
</tr>
<tr>
<td>Event Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>After Update</td>
<td>Called after an item is successfully updated.</td>
</tr>
<tr>
<td>After Shown</td>
<td>Called after an item is shown. Used to tune form representation, such as, hide/show controls, labels, and so on.</td>
</tr>
<tr>
<td>Action Requested</td>
<td>Action executed in context of the form.</td>
</tr>
</tbody>
</table>

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. Navigate to **Application Composer** and select **Common** in the **Application** list.
2. Click **Standard Object**, and then click **Pages** under **Accounts**.
3. Click the **Outlook Pages** tab.
4. Click **Edit Scripts** under **Scripts**.

The List Views page appears.

5. Click the **Create Event Script** icon.
The Edit Event Script page appears.

6. Select **After Shown** from the **Event Type** list.
7. 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()))
}
```

8. 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:

   ```javascript
   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 Custom Objects for Outlook Pages: Explained**

This chapter details steps to add custom objects to Outlook Pages.

To add custom objects to Outlook Pages:

1. Create your custom objects and publish them.

   ✏️ **Note:** Custom 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 customizations.

   For information on creating and publishing custom 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 custom object to Outlook.

3. Test your customizations.
4. Publish this sandbox.

No Outlook pages are defined for custom objects by default. So, after you create a custom object, navigate to Pages, click Outlook Pages, and then click Add to Outlook to configure your object.

Note: 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.

Related Topics
- Defining Objects: Explained
- Publishing Sandboxes: Procedure

Adding Parent Custom Object for Outlook Pages: Procedure

In this example, you create Outlook Pages for the parent custom object called Trouble Ticket so that the Trouble Ticket object is available as a folder in Outlook.
To create Outlook pages for the custom object Trouble Ticket:

1. Navigate to **Application Composer**.
2. Click **Sales** in the **Application** list.
3. Navigate to the custom object **Trouble Ticket**.
4. Click **Pages**, and then click **Outlook Pages**.
5. Click **Add to Outlook**.

![Add to Outlook dialog box](image)

6. 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.

7. Select the change key and natural key fields for the object. You can retain the default selection.

8. Click **Save and Close**.
The Trouble Ticket: Pages page appears and contains links to edit list views, forms, synchronization settings, scripts, and child objects.

### Editing Parent Custom Object List View: Procedure

This topic describes how to edit parent custom 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.

The following figure illustrates how your list view appears on Outlook after you synchronize your changes.
Editing Parent Custom Object Form: Procedure

This topic describes how to edit a parent custom 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 illustrates how your form appears on Outlook after you synchronize your changes.

Adding Custom Child Object: Procedure

This topic provides an example of adding a custom child object on Outlook.

✏ Note: Child objects are not supported for Activity.

In this example, you add a custom child object called **Asset Information** under the Account object in Outlook.

1. Navigate to **Application Composer**.
2. Ensure that **Common** is selected from the **Applications** list.
3. Click **Standard Objects**, and then click **Pages** and **Outlook Pages**.
All links under **Outlook Pages** appear.

4. Click **Edit Asset Information** under **Child Objects**.

   The Edit Asset Information page appears.

5. Click **Add to Outlook**.
The Add to Outlook page appears.

6. Select the change key and natural key fields for the object. You can retain the default selection.
7. Click Save and Close.
The Edit Asset Information page containing all the links appears.

<table>
<thead>
<tr>
<th>Edit Asset Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Views</td>
</tr>
<tr>
<td>Edit List Views</td>
</tr>
<tr>
<td>Form Regions</td>
</tr>
<tr>
<td>Edit Asset Information Form</td>
</tr>
<tr>
<td>Data Synchronization</td>
</tr>
<tr>
<td>Edit Synchronization Settings</td>
</tr>
<tr>
<td>Edit Server Synchronization Filters</td>
</tr>
<tr>
<td>Scripts</td>
</tr>
<tr>
<td>Edit Scripts</td>
</tr>
</tbody>
</table>

Editing Child Custom Object Form: Procedure

This topic describes how to edit a child custom 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**.

![Application Composer](image)

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 custom source object Service Request and the target object Contact.

Prerequisites
Create a custom object called Service Request. See the Adding Parent Custom 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.
2. Select Common from the Application list.
3. Click Relationships under Common Setup.

The Relationships page displays all the existing one-to-many and many-to-many relationships that are already created.

4. Click the Create icon.
The Create Relationship page appears.

5. Select Service Request from the **Source Object** list.
6. Select Contact from the **Target Object** list.
7. 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
8. Enter `ServiceRequestContactRel` in the **Name** field.
9. Enter `ServiceRequestContact` in the **Intersection Object** field.

The intersection object contains the primary identifiers for related records from both source and target objects.

10. 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**.

   The Edit ServiceRequestContact page appears.

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 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.

   The Service Request Child Views page appears.

3. Select **Contacts [ServiceRequestContacts]** by moving it to **Selected Objects** either in Column 1 or 2.

4. Click **Save and Close**.
Contacts [ServiceRequestContacts] appears under 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.

The figure illustrates how the Contacts child region appears in the Service Request form in the Outlook client after you synchronize changes.

Note: Custom 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 customizations 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 customizations within or outside of the Outlook flow. For customizations 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.

> **Note:** You must use the **Generate** button only from within the Application Composer application in which you made the field-level changes. For example, if you made changes to sales lead, you must use the **Generate** button only from within the **Sales** application.

If you have made Outlook-specific customizations 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.
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 machine.
  - **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, Once a Week, or Custom).</td>
</tr>
<tr>
<td>Custom Interval</td>
<td>If you select Custom, specify the interval here in milliseconds.</td>
</tr>
<tr>
<td>Local Synchronization Default Interval</td>
<td>Defines how often Synchronize Local Changes will be run automatically (Once an Hour, Twice a Day, Daily, Once a Week, or Custom).</td>
</tr>
<tr>
<td>Custom Interval</td>
<td>If you select Custom, specify the interval here in milliseconds.</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>
</tbody>
</table>
### Synchronizing and Validating Changes: Explained

After you complete all your customizations 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 customize 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. Customize 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.
20 Partner Relationship Management Extensibility

Overview

Read this chapter to learn about customizing partner management pages in Oracle Sales Cloud. In this chapter, you will learn how to use Page Composer to:

- Customize partner pages used by your employees
- Customize 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 customize these pages:

- Partners landing page
- Create Partner page
- Edit Partner page (see discussion of subtabs below)

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, available in the Sales application.

Partners Landing Page

You can customize 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 customize 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 customize 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 customize 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 customize 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 customize 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 below subtabs are available from the Edit Partner page. Use the below 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
- Customizing Oracle Sales Cloud Simplified Pages: Explained

Customizing Oracle Sales Cloud Partner Pages Using Application Composer: Explained

You can customize a variety of regions on Oracle Sales Cloud partner management pages 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. To access Application Composer, select Application Composer from the Navigator menu, under the Tools category.

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.

Note: You can use and customize desktop UI pages. However, the use of simplified pages, when available, is highly recommended.

How to Make Changes to Partner Management Pages

Here's a quick overview of how to make changes to Oracle Sales Cloud partner management pages:

1. First, identify the page or pages that you want to make changes to.
2. Next, use the table below to determine if a region on the page is extensible.
   - If the region is extensible, identify that region's underlying object.
   - If a region is not listed, then it's not extensible using Application Composer. Instead, use Page Composer to make simple customizations such as hiding or showing a field.
3. In Application Composer, navigate to the desired object.
   
   Note: All partner management objects are listed under the Sales application.

4. Expand the object.
   - Click the Fields node to create custom fields.
   - Click the Actions and Links node to create custom menu actions, toolbar buttons, and links.
5. Next, make your changes visible on the run time partner management pages.
   - Click the Pages node to view the list of configuration pages available for the object. Use the table below to determine which configuration page links let you customize which page regions.
   - Open the desired configuration page and double-click your custom field, action, button, or link to display it at run time.

6. You can now navigate to the run time page to view your changes.

To make only minor user interface changes to partner management pages without creating new objects or fields, use Page Composer instead of Application Composer.

Understanding Which Partner Management Pages Are Extensible

To customize partner management pages, you need to know which pages and regions are extensible, and which objects to select in Application Composer to customize those pages.

This table lists partner management pages and regions in Oracle Sales Cloud, and which objects you must select in Application Composer to customize those pages. For example, the New Partner Registration: Partner Information page is extensible. To create custom fields that you can later add to this page, you must select the Partner object in Application Composer, available within the Sales application.

**Note:** All Oracle Sales Cloud partner management objects are available within the Sales application.

<table>
<thead>
<tr>
<th>Partner Management Page</th>
<th>Partner Management Region</th>
<th>Underlying Business Object</th>
<th>Configuration Page in Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partners Overview page</td>
<td>Summary table region</td>
<td>Partner</td>
<td>Partner: Pages node and then the Edit Summary Table link</td>
</tr>
<tr>
<td>New Partner Registration: Partner Information page</td>
<td>Company Details region</td>
<td>Partner</td>
<td>Partner: Pages node and then the New Partner Registration Enter Company Details link and New Partner Registration Review Company Details link</td>
</tr>
<tr>
<td>Partner Center Profile page</td>
<td>Basic Information, Key Details, and Additional Information regions</td>
<td>Partner</td>
<td>Partner: Pages node and then the Profile: Profile Header Form, Profile: Profile Key Details Form, and Profile: Profile Additional Information Form tree node links</td>
</tr>
<tr>
<td>Partner Center Account Team page</td>
<td>Partner Basic Information region</td>
<td>Partner</td>
<td>Partner: Pages node and then the Profile: Profile Header Form tree node link</td>
</tr>
<tr>
<td>Partner Center Account Team page</td>
<td>Account Team summary table</td>
<td>Partner Account Team</td>
<td>Partner Account Team: Pages node and then the Edit Summary Table link</td>
</tr>
<tr>
<td>Partner Center Enrollments page</td>
<td>Partner Basic Information region</td>
<td>Partner</td>
<td>Partner: Pages node and then the Profile: Profile Header Form tree node link</td>
</tr>
</tbody>
</table>
Customizing Partner Management Dashboard and Pages Using Page Composer: Explained

You can use Page Composer to customize Oracle Fusion Partner Management pages. This includes both external-facing pages used by partners, and internal-facing pages used by your employees.

Customizing External-Facing PRM Pages

To customize external-facing pages, you must have the Channel Partner Portal Administrator privilege.

You can customize the pages at any of the available customization layers, including Site, External or Internal, and Job Role. You can use both the Design and Select modes to:

- Change page layout to any of the eight available layouts. For example, you can change a two-column layout to a three-column layout.
- Add, rename, or remove Partner dashboard subtabs (except the predefined tabs labeled Partner Administrator and Partner Sales Representative).
- Expand or collapse the dashboard Regional pane.
- Add or remove content.
- Show or hide a box, reorder child regions, and change display and style options by editing the dashboard panel box properties.
- Add, remove, and edit regions, links, images, text boxes, movable boxes, and Web pages on dashboards.

<table>
<thead>
<tr>
<th>Partner Management Page</th>
<th>Partner Management Region</th>
<th>Underlying Business Object</th>
<th>Configuration Page in Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partner Center Enrollments page</td>
<td>Summary table region</td>
<td>Program Enrollments</td>
<td>Partner: Pages node and then the Enrollments tree node link</td>
</tr>
<tr>
<td>Edit Enrollment: Specify Agreement Options page</td>
<td>Enrollment Details region</td>
<td>Program Enrollments</td>
<td>Program Enrollments: Pages node and then the Edit Creation Page link</td>
</tr>
<tr>
<td>Partner Center Enrollment Details page</td>
<td>Details region</td>
<td>Program Enrollments</td>
<td>Program Enrollments: Pages node and then the Edit Summary Form link</td>
</tr>
<tr>
<td>Partner Programs page</td>
<td>Summary table region</td>
<td>Partner Programs</td>
<td>Partner Programs: Pages node and then the Edit Summary Table link</td>
</tr>
<tr>
<td>Edit Program page</td>
<td>Summary region</td>
<td>Partner Programs</td>
<td>Partner Programs: Pages node and then the Edit Summary Form link</td>
</tr>
</tbody>
</table>
To customize the remaining external partner pages (Edit Partner Profile, Edit Personal Profile, Partner Landing, and Partner Registration Landing), use the following steps:

1. Navigate to the Partner dashboard.
2. Click on the user name and select *Customize Work Area Pages...* from the Settings and Actions menu.
   
   Page Composer opens.

To customize the partner registration pages (Partner Registration Landing, Partner Registration: Partner Information, Partner Registration: Review and Accept Terms, and Partner Registration Confirmation), use the following steps:

1. In the Partner dashboard, select *View Partner Portal Registration Pages* in the Tasks region.
   
   The Partner Registration Landing Page opens.
2. Click *Register Your Company as a New Partner*.
   
   The partner registration page opens.

You can perform the following customizations:

- Change local area layout (except for Edit Partner Profile and Edit Personal Profile pages).
- Add, rename, or remove Partner dashboard subtabs (except for the predefined tabs for Partner Administrator and Partner Sales Representative job roles).
- Expand or collapse the dashboard Regional pane.
- Add or remove panel boxes to or from the dashboard local area.
- Add predefined content to the dashboard panel.
- Edit dashboard panel box properties: show or hide box, reorder child regions, change display options, and style.
- Add, edit, or remove dashboard components, including regions, links, images, text boxes, movable boxes, and Web pages.

**Customizing Internal-Facing PRM Pages**

To customize the Channel dashboard or any of the internal-facing pages, you must have the Channel Partner Portal Administrator privilege or the Channel Administrator privilege.

To customize the Channel dashboard, use the following steps:

1. Navigate to the dashboard page.
2. Click on the user name and select *Customize Work Area Pages...* from the Settings and Actions menu.

Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Select customization modes. You can perform the following customizations:

- Change the local area layout. For example, you can change a two-column layout to a three-column layout (eight layouts are available).
- Add, rename, or remove Channel dashboard subtabs (except the predefined Channel Manager tab).
- Expand or collapse the dashboard Regional pane.
- Add or remove panel boxes to or from the dashboard local area.
- Add predefined content to dashboard panels.
- Edit dashboard panel box properties: show or hide box, reorder child regions, change display and style options.
- Add, edit, or remove components to or from dashboards, for example, regions, links, images, text boxes, movable boxes, and Web pages.
To customize the Edit Partner Profile and Partner Snapshot pages, you must have either the Channel Partner Portal Administrator privilege or the Channel Administrator privilege.

To access the Edit Partner Profile page, use these steps:

1. Navigate to the Channel dashboard.
2. Select **Review Partners**.

From the Edit Partner Profile page, you can access the Partner Snapshot page. You can launch Page Composer in any of these pages by clicking on the user name and selecting **Customize Work Area Pages...** from the Settings and Actions menu.

Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Select customization modes. You can perform the following customizations:

- Change the local area layout. For example, you can change a two-column layout to three-column layout (eight layouts are available).
- Add, rename, or remove the Channel dashboard subtabs (except for the predefined tab for the Channel Manager job role).
- Expand or collapse the dashboard Regional pane.
- Add or remove panel boxes to or from the dashboard local area.
- Add Resource Library content to dashboard panels.
- Edit dashboard panel box properties: show or hide box, reorder child regions, and change display and style options.
- Add, edit, or remove dashboard components, such as regions, links, images, text boxes, movable boxes, and Web pages.

**Customizing the Partner Portal UI Shell**

You can customize the standard Oracle Fusion user interface (UI) shell for the application’s external facing pages when you are signed in as a user with the Channel Partner Portal Administrator job role. The UI shell is the header and footer area which appears in all application pages.

To customize the UI shell:

1. Navigate to the Partner Portal.
2. From the Tasks region, select **Update Partner Portal UI Shell**.

This customization task opens Page Composer in the Source View, which is enabled only for this task in Oracle Sales Cloud.

You can perform the following customizations:

- Modify the user interface shell header. For example, you can add new content, hide or show global links, and hide and show menu items.
- Modify the user interface shell footer by adding, editing, or hiding links.
- Replace the branding logo.
- Replace the branding text.
- Change the menu rendering. You can use either the Oracle Fusion Navigator or a tabbed-style menu.

**Related Topics**

- Customizing Oracle Sales Cloud Pages Using Page Composer: Explained
21 Territory Management Extensibility

Extending Simplified Pages for Territories: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages for sales territories.

✏️ Note: The ability to customize desktop pages is applicable if you have implemented Oracle Sales Cloud in Release 9 or earlier. For all implementations, the use of simplified pages is highly recommended.

Use Application Composer to customize 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, available in the Sales application.

Territories Landing Page

You can extend the Territories landing page. For example, you can hide standard fields or show custom fields. To customize 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 customize the Create Territories page. For example, you can hide standard fields, make a field required, add a default value, and show custom fields. To customize 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 left side of the page. You can customize most of these subtabs. You can also add custom 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 customize 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
   - Coverage
     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.

Customizing Oracle Sales Cloud Territory Management: Explained

You can customize 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 Tools section, select Application Composer.

Understanding Which Territory Pages Are Extensible

To customize Territory pages, you need to know which pages and regions are extensible, and which objects to select in Application Composer to customize those pages.
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 customize 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. Select the Sales application on the main 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 customize.

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</td>
<td>Select the <strong>Pages</strong> node for this object, then click <strong>Edit Summary Form</strong>.</td>
<td>Overview page</td>
<td>Details region for a selected territory, Summary region on the Profile tab</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>
22 Quota Management Extensibility

Overview

Read this chapter to understand how you customize and extend Oracle Sales Cloud Quota Management using Application Composer.

This chapter covers:

- Which pages and regions are extensible
- How you expose custom fields and objects in Quota Management application

Customizing Oracle Sales Cloud Quota Management: Explained

You can customize a variety of regions in Oracle Sales Cloud Quota Management using Application Composer. Application Composer lets you create custom fields, actions, and links, which you then add for display in the quota management user interface pages. You can also create custom child objects, which you can add to certain pages as subtabs or tree nodes. This topic addresses the ability only of desktop pages to be configured.

To access Application Composer:

1. Go to the Navigator menu.
2. In the Tools section, select Application Composer.

Understanding Which Sales Quota Pages Are Extensible

To customize Sales Quota pages, you need to know which pages and regions are extensible, and which objects to select in Application Composer to customize those pages.

To customize Sales Quota pages, find the Sales Goal object.

This table lists Sales Quota pages and regions, and the related objects that you can access in Application Composer to customize those pages. For example, the Products tab on the Edit Sales Goal page is extensible. To create custom fields that you can later add to the Products tab, you must select the Sales Goal Product object in Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Sales Quota Page</th>
<th>Sales Quota Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview page</td>
<td>Search and results regions</td>
<td>Sales</td>
<td>Sales Goal</td>
</tr>
<tr>
<td>Create Sales Goal page</td>
<td>Create Sales Goal region (click on New to go to Create page)</td>
<td>Sales</td>
<td>Sales Goal</td>
</tr>
</tbody>
</table>
Adding Your Changes to the User Interface Pages

To add custom fields to the Sales Quota regions listed in the table in the previous section:

1. Select the Sales application on the main Overview page.
2. In the Standard Objects tree, select the Sales Goal 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 Sales Quota region that you want to customize.
6. Add your custom fields to the desired Sales Quota regions.

This table indicates which Sales objects populate which Sales Quota pages and regions, as well as the 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 Sales Quota Page</th>
<th>Related Sales Quota Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Goal</td>
<td>Select the Pages node for this object, then click Edit Local Search.</td>
<td>Manage Sales Goal page</td>
<td>Manage Sales Goal, Search region</td>
</tr>
<tr>
<td>Sales Goal</td>
<td>Select the Pages node for this object, then click Edit Summary Table.</td>
<td>Manage Sales Goal page</td>
<td>Manage Sales Goal, Search Results region</td>
</tr>
<tr>
<td>Sales Goal</td>
<td>Select the Pages node for this object, then click Edit Creation Page.</td>
<td>Create Sales Goal page</td>
<td>Create Sales Goal, Summary region</td>
</tr>
<tr>
<td>Sales Goal</td>
<td>Select the Pages node for this object, then click Edit Summary Form.</td>
<td>Edit Sales Goal page</td>
<td>Edit Sales Goal, Summary region</td>
</tr>
<tr>
<td>Sales Goal Product (child of Sales Goal)</td>
<td>Select the Pages node for the Sales Goal object, select the</td>
<td>Edit Sales Goal page</td>
<td>Edit Sales Goal, Products tab</td>
</tr>
<tr>
<td>Business Object</td>
<td>Configuration Page in Application Composer</td>
<td>Related Sales Quota Page</td>
<td>Related Sales Quota Region</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>Products subtab row, and click the <strong>Edit</strong> icon to navigate to the Edit Subtab: Child or Related Object page.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
23 Sales and Opportunity Management Extensibility

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 customize sales and opportunity management objects using Page Composer.

Related Topics
- Customizing the Sales Dashboard Using Page Composer: Worked Example

Extending Simplified Pages for Opportunities: Explained

Using Application Composer, you can change many items that appear in the simplified set of pages for opportunities. Use Application Composer to customize 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 custom fields of all types.
- Set fields as required.
• Set field default values.
• Reorder columns.
• Change column labels.
• Add custom actions and buttons.

High-Level Steps to Customize Opportunity Pages
The following are the high-level steps to make changes to the opportunity simplified UI pages.

1. Activate a sandbox.
2. Navigate to the Application Composer, available under the Tools menu.
3. In Application Composer, select Sales from the Application list.
4. Expand the Opportunity object tree.
5. In the Opportunity tree, click the Pages link.
6. Ensure that the Simplified Pages tab is active.
7. In the region that corresponds to the page you are customizing, duplicate the standard layout and edit the resulting custom layout. For example, in the Details Page Layout region, duplicate the standard layout and then edit the resulting custom layout. See the Customizing Opportunity Pages section in this topic for details about which regions in the Simplified Pages tab correspond to which opportunity pages.
8. In the new custom layout row, the Active check box will be automatically checked, indicating that the new custom 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.

9. Save your changes.
10. Publish the sandbox.

Customizing Opportunity Pages
The following table shows the opportunity pages or subtabs that can be customized, along with the navigation steps to the areas where you make the customizations.

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>
<tbody>
<tr>
<td>Landing (list) page</td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Landing Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td>Create Opportunity Page</td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Creation Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td>Edit Opportunity Page, Summary region</td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Opportunity Page</td>
<td>Navigation</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Ensure the Summary subtab is active.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Summary region.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Products table</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Opportunity Revenue object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Edit the Edit Revenue Table region.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Contacts subtab</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Opportunity Contact object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Contacts subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Contacts subtab.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Team subtab, Team view</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Opportunity Team Member object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Team subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Fuse Opportunity Team region.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Team subtab, Territories view</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Sales Territory object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Team subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Territories region.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Partners subtab</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Opportunity Partner object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Partners subtab.</td>
</tr>
<tr>
<td></td>
<td>5. Edit the Partners subtab.</td>
</tr>
<tr>
<td></td>
<td><strong>Edit Opportunity Page, Deal Registrations subtab</strong></td>
</tr>
<tr>
<td></td>
<td>Optionally, first add custom fields to the Opportunity Deal object.</td>
</tr>
<tr>
<td></td>
<td>1. Expand the Opportunity object.</td>
</tr>
<tr>
<td></td>
<td>2. Click the Pages link.</td>
</tr>
</tbody>
</table>
In the Details Page Layouts region, duplicate the standard layout and edit the resulting custom layout.

4. Click the **Deal Registrations** subtab.

5. Edit the Deal Registrations subtab.

**Edit Opportunity Page, Activities subtab**

You must use the Activity object in Application Composer to extend this subtab.

1. Navigate to the **Activity** object, under the Sales application.
2. Click **Pages** and ensure the **Simplified Pages** tab is active.
3. In the **Embedded Activity Summary Layout region** at the bottom of the page, duplicate the standard layout and edit the resulting custom layout.

**Edit Opportunity Page, Notes subtab**

You must use the Notes object in Application Composer to extend this subtab.

1. Navigate to the **Notes** object, under the Common application.
2. Click **Pages** and ensure the **Simplified Pages** tab is active.
3. Duplicate the standard layout and edit the resulting custom layout.

The Notes list page is not extensible.

**Edit Opportunity Page, Leads subtab**

Optionally, first add custom fields to the Sales Lead object.

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 custom layout.
4. Click the **Leads** subtab.
5. Edit the Fuse Lead Overview table.

The following are not extensible:

- Quotes and Orders
- Assessments
- Conversations

**Related Topics**

- Customizing Oracle Sales Cloud Simplified Pages: Explained

---

**Extending Simplified Pages for Opportunities: Worked Example**

This example demonstrates how you can customize simplified pages for Opportunities using Application Composer.
Extending Simplified Pages for Opportunities Using Application Composer

1. Select the Sales application on the main Overview page.
2. In the object tree, select a standard object that includes a set of simplified pages, such as Opportunity.

   Application Composer

<table>
<thead>
<tr>
<th>Application</th>
<th>Sales</th>
</tr>
</thead>
</table>

   ▲ Objects
   ▲ Custom Objects
     ▲ Quote
     ▲ Sales Tool
   ▲ Standard Objects
     ▲ Forecast Item
     ▲ Forecast Territory Details
     ▲ Opportunity
     ▲ Opportunity Contact
     ▲ Opportunity Reference
     ▲ Opportunity Revenue
     ▲ Opportunity Team Member

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 customize the simplified pages that are available for the selected object. You can show or hide fields, rearrange fields, and add custom fields.
6. For example, if you want to add the field 'Budget Amount' to the Create Opportunity page, in the Simplified Pages landing 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.

   Note: Changes that you make to a simplified page are not automatically replicated on the object’s corresponding desktop page.
9. Click Save and Close.

Related Topics
- Extending Simplified Pages: Overview
Customizing Fields in Simplified UI Pages Using Page Composer: Worked Example

Use this example to learn how to customize fields in Oracle Sales Cloud simplified UI pages using Page Composer. In this example, you learn how to customize the **Primary Competitor** field on the Edit Opportunities page for users with the Sales Representative job role. Customizations covered include hiding the field, make it required, and changing its label.

Before you customize 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 customizations 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.**

   ![](image)

   The Manage Sandboxes window appears.
2. If no sandbox is listed or if you want to keep your customizations separate from other customizations, 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.
Customizing the Field

1. To customize a field in the Edit Opportunity page, click **Opportunities** to display the list of opportunities.

2. Click the user name at the top right of the page, and select **Customize User Interface** from the menu to open Page Composer.

   **Tip:** When you are customizing simplified 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 customize.
The Customize User Interface window appears.

3. Select the layer that you want to customize. In this example, you want to customize 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 border appears at the top of the page indicating that you are editing the page with the Design mode selected.

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 customize, 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.

8. Select **Edit Component**.

The Component Properties window appears. Not every field supports all of customizations, so the content of this window varies.
The numbers in the following image refer to the customizations described below.

![Component Properties: Primary Competitor](image)

9. Customize the field as 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 the down arrow to the right of the **Label** field (4)
ii. Select **Select Text Resource** from the menu.

The Select Text Resource window appears.

![Select Text Resource window](image)
iii. Enter a name in the **Key** field. You can use this name to search for this label when you customize other pages.

iv. Enter the new field label in the **Display Value** field.

v. Click **OK**.

10. After you complete your customizations in the Component Properties window, click **Apply** and click **OK**.

Your customizations are visible on the page while you are in Page Composer.

11. Click **Close** to close Page Composer.

![Edit: User Interface](image)

**Note:** Your customizations are no longer visible on the page because you are not signed in as a salesperson.

12. You can review the customization on the page while still in the sandbox by signing out and signing in again as a salesperson. To make the customizations permanent, you must publish the sandbox.

---

**Customizing Sales Pages: Explained**

You can use Application Composer to customize a variety of regions in the Oracle Sales Cloud 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.

**Extensible Competitor Pages and Navigation**

To customize the Competitors pages, you need to know which pages and regions are extensible and which objects to select in Application Composer to customize those pages.

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 simplified UI pages. 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. |
Creating Custom Competitor Fields

If you want to have custom fields that you can add to the Competitors pages, you must create them first. The following procedure shows how to add a custom text field. For more details about adding fields, see the topics on adding fields using Application Composer.

1. Sign in as the sales administrator or as a setup user.
2. Navigate to **Tools - Application Composer**.
   
The Application Composer overview page appears.
3. Select the **Sales** application from the Application choice list.
4. In the Objects tree, under Standard Objects, expand the object tree for the object you are customizing. 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.
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**.

Customizing Competitor Pages

Use the following procedure to customize Competitors pages or regions.

1. Sign in as the sales administrator or as a setup user.
2. Navigate to **Tools - Application Composer**.
   
The Application Composer overview page appears.
3. Select the **Sales** application from the Application choice list.
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 Competitors pages for the web-based application are housed under the Desktop Pages tab. Click the **Desktop Pages** tab.

---

**Note:** You must work in an active sandbox to be able to access the customization pages.
7. Make the required customization changes. For additional navigation information, refer to the section in this topic, Extensible Competitor Pages and Navigation.
8. Save your changes.

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 will see two fields for Primary Competitor. The names you see in the list are display names only; each has an underlying name. To see the underlying (actual) 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 Simplified Pages for Households: Explained

Customizing Opportunity Pages: Explained
You can customize many opportunity pages and regions using Application Composer. For example, you can modify pages and create custom fields and objects, which you then make available in the run time opportunity application.

Note: This topic is geared toward customizing opportunity desktop pages, although the opportunity-related objects that support customization are generally the same between the two UIs. There may be some exceptions, and these should be apparent as you work within Application Composer. While you can use and customize desktop UI pages, the use of simplified pages, when available, is highly recommended. To understand what opportunity-related objects support customization in the simplified UI and how to configure the set of simplified pages for these objects, see the related topics on customizing simplified pages for opportunities.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools category. The opportunity areas that can be customized are listed under Sales in the Application list of values.

Understanding Which Opportunity Pages Are Extensible
To customize opportunity pages, you need to know which pages and regions are extensible and which objects to select in Application Composer to customize those pages.

The Sales objects that are associated with opportunity pages are:

- Opportunity: This is the parent object.
- Opportunity Contact: This is a child object of the Opportunity object, used for sales contacts.
- Opportunity Deal: This is a child object of the Opportunity object, used for partner deal registrations.
- Opportunity Partner: This is a child object of the Opportunity object, used for partner management.
- Opportunity Reference: This is a child object of the Opportunity object, used for sales reference customers. Reference customer functionality is not available in the simplified UI.
- Opportunity Revenue: This is a child object of the Opportunity object, used for opportunity products areas.
- Opportunity Team Member: This is a child object of the Opportunity object, used for sales team member areas.

Note: To make only minor UI changes to opportunity pages without creating new objects or fields, use Page Composer instead of Application Composer.
The following table lists the pages and regions that you can access in Application Composer to customize opportunity pages.

<table>
<thead>
<tr>
<th>Opportunity Page</th>
<th>Opportunity Region</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunity landing page</td>
<td>Opportunity landing (list view) table</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Create Opportunity page</td>
<td>Create Opportunity page</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Summary region, also called the header region. This is the region above the details region that is available by selecting the Show More option.</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Detail region (expand the Show More option)</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Revenue table</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Revenue table detail stamp (the region uncovered when you expand the revenue item row)</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Contacts summary table</td>
<td>Opportunity Contact (child object of opportunity)</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Contacts summary table actions</td>
<td>Opportunity Contact (child object of opportunity)</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Opportunity Team summary table</td>
<td>Opportunity Resource (child object of opportunity)</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Opportunity Team summary table detail stamp (the region uncovered when you expand the team member row)</td>
<td>Opportunity Resource (child object of opportunity)</td>
</tr>
</tbody>
</table>

To create custom fields that you can later add to the Edit Opportunity page, select the Opportunity object in the Application Composer and create your custom fields.

**Adding Your Changes to the Run Time Application**

To add custom fields to the opportunity regions listed in the table above, first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the desired opportunity regions.

*Note:* You must work in an active sandbox to be able to access the customization pages.

To access the opportunity configuration pages:

1. Sign in as the sales administrator or as a setup user.
2. Navigate to **Tools - Application Composer**.
3. Select the Sales application from the Application choice list.
4. In the Objects tree, under Standard Objects, expand the object tree for the object you are customizing. For example, expand the Opportunity tree.
5. Expand the element of the object you are customizing. For example, click the Pages node for the object. A page with tabs for each UI type appears.
6. Click the tab for the UI you are customizing. For example, select Desktop Pages.

>Note: For a few objects, you can only customize fields, not pages or regions. For those objects, a Pages node will not be available.

7. Click the link of the region you are customizing. For example, click the link for the summary table.
8. Make the required customization changes.

Navigating to Configuration Pages

The following table shows the extensible pages and regions for the Opportunity object and its child objects, along with the navigation to the configuration pages.

<table>
<thead>
<tr>
<th>Business Object and Related Page or Region</th>
<th>Navigation in Application Composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Opportunity landing (list) page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Summary Table link.</td>
</tr>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Create Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td></td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Creation Page link.</td>
</tr>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>• Summary region (also called header region), Default area</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Summary Form link.</td>
</tr>
<tr>
<td></td>
<td>5. Use the Configure Default Summary region.</td>
</tr>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>• Summary region (also called header region), Details area (available by expanding the Show More area)</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Summary Form link.</td>
</tr>
<tr>
<td></td>
<td>5. Use the Configure Detailed Summary region.</td>
</tr>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>• Revenue Items summary table</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Revenue Table link.</td>
</tr>
<tr>
<td></td>
<td>5. Use the Configure Summary Table region.</td>
</tr>
<tr>
<td>• Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>• Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Business Object and Related Page or Region</td>
<td>Navigation in Application Composer</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Revenue Items table, detail stamp</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Revenue Table link.</td>
</tr>
<tr>
<td></td>
<td>5. Use the Configure Summary Table: Detail Stamp region.</td>
</tr>
<tr>
<td>Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Contact summary (list view) table</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Find the Opportunity Detail Tab region.</td>
</tr>
<tr>
<td></td>
<td>5. Find Opportunity Contact in the table and edit it.</td>
</tr>
<tr>
<td></td>
<td>6. Use the Configure Summary Table region.</td>
</tr>
<tr>
<td>Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Contact summary (list view) table</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td>buttons and actions</td>
<td>4. Find the Opportunity Detail Tab region.</td>
</tr>
<tr>
<td></td>
<td>5. Find Opportunity Contact in the table and edit it.</td>
</tr>
<tr>
<td></td>
<td>6. Use the Configure Summary Table: Buttons and Actions region.</td>
</tr>
<tr>
<td>Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Opportunity Team tab</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td>Team summary (list view) table</td>
<td>4. Find the Opportunity Detail Tab region.</td>
</tr>
<tr>
<td></td>
<td>5. Find Opportunity Team in the table and edit it.</td>
</tr>
<tr>
<td></td>
<td>6. Use the Configure Summary Table region.</td>
</tr>
<tr>
<td>Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Opportunity Team tab</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td>Team summary (list view) table, detail stamp</td>
<td>4. Find the Opportunity Detail Tab region.</td>
</tr>
<tr>
<td></td>
<td>5. Find Opportunity Team in the table and edit it.</td>
</tr>
<tr>
<td></td>
<td>6. Use the Configure Summary Table: Detail Stamp region.</td>
</tr>
<tr>
<td>Opportunity</td>
<td>1. Select the Opportunity object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Opportunity Team tab</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td>Team summary (list view) table, buttons and actions</td>
<td>4. Find the Opportunity Detail Tab region.</td>
</tr>
<tr>
<td></td>
<td>5. Find Opportunity Team in the table and edit it.</td>
</tr>
<tr>
<td></td>
<td>6. Use the Configure Summary Table: Buttons and Actions region.</td>
</tr>
<tr>
<td>Opportunity Contact</td>
<td>1. Select the Opportunity Contact object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Fields link.</td>
</tr>
<tr>
<td>Contact detail view (fields only)</td>
<td></td>
</tr>
<tr>
<td>Opportunity Revenue</td>
<td>1. Select the Opportunity Revenue object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Pages link.</td>
</tr>
<tr>
<td>Opportunity Revenue Items summary (list view) table</td>
<td>3. Click the Desktop Pages tab.</td>
</tr>
<tr>
<td></td>
<td>4. Click the Edit Summary Table link.</td>
</tr>
<tr>
<td>Opportunity Team Member</td>
<td>1. Select the Opportunity Team Member object.</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>2. Click the Fields link.</td>
</tr>
<tr>
<td>Opportunity team member detail view (fields only)</td>
<td></td>
</tr>
</tbody>
</table>
Duplicate Display Names for Fields

When you configure opportunity pages, in the Available Fields list, some field display names appear to be duplicates. For example, you will see two fields for Account. The names you see in the list are display names only; each has an underlying internal name. To see the underlying internal name of the field, hover your mouse pointer over the field name. The hover text displays the actual name of the field.

Related Topics

- Customizing and Extending Oracle Sales Cloud Applications: Overview
- Extending Simplified Pages for Households: Explained
- Customizing Oracle Sales Cloud Pages Using Page Composer: Explained

Customizing Your Object to Validate Assessments: Worked Example

Use this example to learn how to customize 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.
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.

```groovy
def optyStatusCode = getAttribute('StatusCode')
if (optyStatusCode == 'WON') {
    def id = getAttribute('OptyId')
    def vo = newView('AssessmentVO')
    def vc = vo.createViewCriteria()
    def vcr = vc.createRow()
```
```python
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) {
    vc.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.

### Customizing the Desktop UI Sales Dashboard and Pages Using Page Composer: Explained

You can use Page Composer to customize the Sales dashboard, the Opportunity work area, and related pages when you sign in as a user with sales administrator privileges.

### Customizing the Desktop UI Sales Dashboard

To customize the Sales dashboard, you:

1. Select **Sales Dashboard** from the Navigator menu.
2. Click on the user name and select **Customize Home Pages** from the menu.

In the Page Composer Design mode you can:

- Change the page layout to any of the eight available layouts. For example, you can change a two-column layout to a three-column layout.
• Add, rename, or remove Sales dashboard subtabs (except for the predefined Home tab).
• Expand or collapse the dashboard Regional pane (or move the splitter location).
• Add or remove panel boxes to or from the dashboard local area.
• Add predefined content to the dashboard panel.
• Edit the dashboard panel box properties. You can show or hide a box, reorder child regions, display options, and change styles.
• Add, edit, or remove different components, including regions, links, images, text boxes, movable boxes, and Web pages.

In the Page Composer Select mode, you can change the order of links in the Tasks regional bin.

Customizing Sales Pages in the Desktop UI

To customize sales pages:

1. Navigate to the page that you want to customize (for example, the Edit Opportunity and Revenue Item pages).
2. Launch Page Composer in any of these pages by clicking on the user name and selecting the first entry under the Administration heading. The task name varies depending on the object you are customizing. For example, for opportunities, you select Customize Opportunities Pages...

You can perform any of the following customizations in the Select mode:

• Hide or show a field.
• Change a field label.
• Make a field required or not.
• Make a field read-only or updatable.
• Reorder fields in a form.
• Reorder table columns.
• Hide or show table columns.
• Set table column width and minimum width in percent or pixels.
• Enable or disable column sorting.

Related Topics

• Personalizing Dashboards, Transactional Pages, and Landing Pages Using Page Composer: Overview
• Customizing Oracle Sales Cloud Pages Using Page Composer: Explained

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 customizations.
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 customization sandboxes following your organization’s established processes. As a best practice, your organization might want you to first make custom 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 customization 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.

![Component Properties: Win Probability (%)](image)

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 customization 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 customization.
4. Click the **Publish** button to commit your changes.
Verifying Your Changes

Verify your customizations 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.

Customizing the Sales Dashboard in the Desktop UI: Procedure

Use this procedure to add content or otherwise customize to the sales dashboard in the desktop UI.

Customizing the Sales Dashboard Page

1. In the Navigator, select the **Sales Dashboard** under the **Sales** heading.
2. Make sure you are working in an active sandbox: A sandbox name should appear to the left of the icons in the toolbar if you are.
3. Click on the user name at the top right hand corner of the page and select **Customize Home Pages** from the Settings and Actions menu.
4. In the Customize Home pages window, select the customization layer:
   - To customize the dashboard for all users, select **Site**.
   - To customize the dashboard for internal or external users, select **External Or Internal** and specify which one you want. Internal users are your employees. External users are your partners.
   - To customize the dashboard for a specific job role, select **Job Role** and then select the job role.
5. Click **OK**.
The page opens in the Page Composer Design mode as shown in the following image. Numbers are added to identify Design mode features.

- A bar appears at the top of the page indicating that you are in editing the page in Page Composer (1).
- You can change the column layout of the page by clicking **Change Layout** (2).
- You can add a new tab (3).
- You can move a region, by grabbing its toolbar (4) and dragging it to a different position.
- You can remove a region by clicking the close icon at the top right hand corner of the region.
- You can add content by clicking **Add Content**.
You add a new region by clicking on one of the icons to the right of the Add Content button. These icons specify where in the page the new region will be created.

6. Click **Save** after each customization.
7. Click **Save and Close** when you are done.
8. Click **Close** to exit from Page Composer.

The dashboard reflects your customizations immediately, but they will not be seen by users until you publish the sandbox.
24 Sales Forecasting Extensibility

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.

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 customize these pages:

- Sales Forecasts landing page
- Forecast Details page

  - Products subtab
  - Forecast Items subtab

When you customize the Forecast Details page, you'll 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 customize 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 customize 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 extend the region that appears at the bottom of the card view. You can extend 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 extensible 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 extend 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 extend 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 Customize the Sales Forecasts Simplified Pages**

Before customizing pages, create custom fields and make other changes using the Forecast Territory Details standard object.

To customize the simplified pages for sales forecasts using Application Composer:

1. Select the Sales application on the main 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 customize 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.</td>
</tr>
<tr>
<td></td>
<td>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>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.</td>
</tr>
<tr>
<td></td>
<td>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>
</tbody>
</table>
Customizing Oracle Sales Cloud Sales Forecasting Pages: Explained

You can customize a variety of regions in Oracle Sales Cloud Sales Forecasting using Application Composer. You can create custom fields and objects, which you then add for display in the runtime Sales Forecasting application.

This topic addresses 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 Tools section, select Application Composer.

Understanding Which Sales Forecasting Pages Are Extensible

To customize Sales Forecasting pages, you need to know which pages and regions are extensible, and which objects to select in Application Composer to customize those pages.

The Oracle Sales objects that are associated with Sales Forecasting pages are:

- Forecast Item
- Forecast Territory Details

This table lists Sales Forecasting pages and regions, and the related objects that you can access in Application Composer to customize those pages. For example, the Forecast Items tab on the Edit Forecast page is extensible. To create custom fields that you can later add to the Forecast Items tab, you must select the Forecast Item object in Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Sales Forecasting Page</th>
<th>Sales Forecasting Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Forecast page</td>
<td>Territory Details region (expand the Show More option)</td>
<td>Sales</td>
<td>Forecast Territory Details</td>
</tr>
<tr>
<td></td>
<td>Forecast Items tab</td>
<td>Sales</td>
<td>Forecast Item (child of Forecast Territory Details)</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Forecast Items tab by accessing the appropriate Application Composer configuration page, listed in the following section.
Note: To make only minor user interface changes to Sales Forecasting 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 Sales Forecasting regions listed in the table in the previous section: first create your custom fields using Application Composer. Next, use Application Composer’s configuration pages to add those custom fields to the desired Sales Forecasting regions. You access the configuration pages in Application Composer from the Pages node under the Forecast Territory Details object.

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Forecast Territory Details 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 Sales Forecasting region that you want to customize.

This table indicates which Sales objects populate which Sales Forecasting 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 Sales Forecasting Page</th>
<th>Related Sales Forecasting Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forecast Item (child of Forecast Territory Details)</td>
<td>Select the Pages node for the Forecast Territory Details object, select the Forecast subtab row, and click the Edit icon to navigate to the Edit Subtab: Child or Related Object page.</td>
<td>Edit Forecast page</td>
<td>Forecast Items tab</td>
</tr>
<tr>
<td>Forecast Territory Details</td>
<td>Select the Pages node for this object, then select the Desktop Pages tab and click Edit Summary Form.</td>
<td>Edit Forecast page</td>
<td>Territory Details region</td>
</tr>
</tbody>
</table>

Note: To view your custom fields in the Territory Details region, you must expand the Show More option which appears above the Forecast Items tab.

Related Topics

- Defining Pages: Explained
- Extending Simplified Pages for Households: Explained
25 An End-to-End Configuration Example

Overview

This chapter contains examples of end-to-end configurations that you can do in Oracle Sales Cloud.

In this chapter, you’ll learn:

- How to create a sandbox
- How to create and extend an object, and add child or related objects
- How to enforce business policies using advanced field and object properties
- How to create user interface pages, and add subtabs and tree nodes
- How to test security settings for custom objects
Creating a Sandbox: Example

This example will show you how to create a sandbox.

Creating a Sandbox

1. Click your user name in the global area, then select Manage Sandboxes under Administration.

   The Manage Sandboxes window appears, listing the available sandboxes in your environment.

2. Click the New (plus sign) icon.

3. In the Create Sandbox window, enter a name in the Sandbox Name field, then click Save and Close.

4. To activate the sandbox, select the sandbox that you just created, then click the Set as Active button.

   Whenever you are logged in to the application and working in a session sandbox, the upper left part of your window shows the session sandbox name. You can work in the sandbox and perform typical extension functions. Other users can't see what you've done until you publish your sandbox. Hover your mouse over the name of your sandbox to see more details about it.

   Note: If you sign out and sign in again as the same user, you will still be in the same sandbox. The sandbox you're working in is a part of your user profile information.
5. Click your sandbox name in the upper left part of the window. As you make extensions in your sandbox, various XML files in the MDS repository are changed. In this case, we’re taking a quick tour of sandboxes, and we have not made any changes, so there aren’t any files showing. If you make changes while in your sandbox, this is a way to see what all those changes are, exactly which XML files in the MDS repository have been changed, and the layers of the changes.

6. Click the More link. The Sandbox Details window appears.

7. Click the Close button. You can exit the sandbox by clicking the sandbox name and clicking Exit Sandbox. In this configuration example, however, we want to work in a sandbox, so let’s move on to the next step in our example, creating a custom object.

Related Topics
- Using Sandboxes: Overview

Creating an Object: Example

This example demonstrates how to create an object. Here, an Oracle Sales Cloud administrator at Pinnacle Corporation uses Application Composer to create a help request custom object and work area. This will let end users at Pinnacle Corporation log and track help requests from their customers.

Creating the Help Request Object

1. Sign in to Oracle Sales Cloud and click the Navigator link.
2. Under **Tools**, click **Application Composer**
3. Select **Sales** from the **Application** list.
   The object navigation tree shows both the standard and custom objects available in the Sales application container.
4. Click the **Create** icon next to **Custom Objects**.
5. Enter the following information in the indicated fields, pressing the Tab key after each.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Notes</th>
<th>Text to Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Label</td>
<td>This is the name the users will see.</td>
<td>Help Request</td>
</tr>
<tr>
<td></td>
<td>Press the Tab key after entering your text to populate the rest of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the required fields with the text you enter.</td>
<td></td>
</tr>
<tr>
<td>Plural Label</td>
<td>The pluralized version of the display label.</td>
<td>Help Requests</td>
</tr>
<tr>
<td>Record Name Label</td>
<td>The name of the field that identifies each instance of the object.</td>
<td>Abstract</td>
</tr>
<tr>
<td></td>
<td>In the help request object, this will be the abstract, where users</td>
<td></td>
</tr>
<tr>
<td></td>
<td>enter a brief summary of the issue.</td>
<td></td>
</tr>
<tr>
<td>Object Name</td>
<td>The internal name of the object.</td>
<td>HelpRequest</td>
</tr>
<tr>
<td>Other fields</td>
<td>Leave as is</td>
<td></td>
</tr>
</tbody>
</table>

6. Click **OK**.
7. Click the **Expand** button next to the Custom Objects tree item.
   The list of custom objects now contains the help request object you created.

### Adding Fields to the Help Request Object

1. Click the **Expand** icon next to the Help Request tree item.
   The icons represent what you can create for your object including fields, pages, actions, and links. You’ll start by creating these fields for your object:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A text field for the description of the issue.</td>
</tr>
<tr>
<td>Type</td>
<td>Classification of the issue in the request: Hardware, Software, or Network.</td>
</tr>
<tr>
<td>Status</td>
<td>Help Request status that can be either Open or Closed.</td>
</tr>
<tr>
<td>Company</td>
<td>A field you will use to select the partner company that is experiencing the</td>
</tr>
<tr>
<td></td>
<td>problem and thus requesting help.</td>
</tr>
</tbody>
</table>

2. Under Help Request, click the **Fields** link.
The Fields page displays.

3. You'll start by creating the Description field. Click the Create a custom field button.

### Fields

<table>
<thead>
<tr>
<th>Custom</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>View</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display Label</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>To add a custom field, click Create.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Select Field Type window appears.

4. Click the Long Text option, then click OK.

5. Enter the following values in the indicated fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Label</td>
<td>Description</td>
</tr>
<tr>
<td>Display Width</td>
<td>50</td>
</tr>
</tbody>
</table>

6. Click the Save and Close button.

The application returns you to the Fields page and you are ready to create your second field: Type.

7. Click the Create a custom field button.

8. Click the Choice List (Fixed) option, then click OK.

9. Enter "Type" in the Display Label field.

10. In the List of Values region, you'll create a Lookup Type with the values users select in this field at run time. Click the Create a New Lookup Type button.

### List of Values

Configure the list of values you want to display in the choice list. Click the search.

<table>
<thead>
<tr>
<th>Lookup Type</th>
<th></th>
<th><img src="edit.png" alt="Edit" /></th>
<th><img src="save.png" alt="Save" /></th>
</tr>
</thead>
</table>

The Create Lookup Type window appears.

11. Enter the following information in the indicated fields:
You're now ready to enter the values users will select (Hardware, Software, and Network) in the Lookup Codes region.

12. Click the Create Lookup Code icon.
13. Enter the following data, clicking Create Lookup Code before entering each new row of data:

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Lookup Code</th>
<th>Display Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Hardware</td>
<td>10</td>
</tr>
<tr>
<td>Software</td>
<td>Software</td>
<td>20</td>
</tr>
<tr>
<td>Network</td>
<td>Network</td>
<td>30</td>
</tr>
</tbody>
</table>

14. Click Save.

The application returns you to the Create Fixed Choice List page. Notice that your lookup type is linked to the Type field you are creating.

15. Click Save and Close.

The application returns you to the Fields page. You're now ready to create the third field for your object: Status.

16. Click the Create a custom field icon.
17. Click the Choice List (Fixed) option, then click OK.

The Create Fixed Choice List page appears.

18. Enter "Status" in the Display Label field.
19. In the List of Values region, click the Create a New Lookup Type button.

The Create Lookup Type window appears.

20. Enter "Help Request Status" in the Meaning field.
21. Enter "HR_STATUS" in the Lookup Type field.
22. Now you'll add the two Status values, Open and Closed, in the Lookup Codes region. Click the Create Lookup Code icon.
23. Enter the following values, clicking the Create Lookup Code icon between each set:

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Lookup Code</th>
<th>Display Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Open</td>
<td>10</td>
</tr>
<tr>
<td>Closed</td>
<td>Closed</td>
<td>20</td>
</tr>
</tbody>
</table>
24. Click **Save**.
   The application returns you to the Create Fixed Choice List page.

25. In the Default Value region, you will specify Open as the status of the help request when it is created. Click the dropdown list next to **Fixed Value**.

26. Select the **Open** list item.

27. Click **Save and Close**.
   The application returns you to the Fields page.
   You’re now ready to create the fourth field for your object: Company. Using this field, you will be able to associate one of the partners in your database with the help request.

28. Click the **Create a custom field** button.

29. Click the **Choice List (Dynamic)** option, then click **OK**.
   The application displays the Create Dynamic Choice List: Basic Information page.

30. Enter “Company” in the **Display Label** field.

31. Click the **Next** button.
   The application displays the Create Dynamic Choice List: List of Values page. You’ll use the List Data Source region to specify which of your objects will populate the **Company** field list of values.

32. Select **Partner** from the **Related Object** list.
Now you’ll select the partner attribute you want displayed in the list.

33. Select **Name** from the **List Selection Display Value** list.

   ✨ **Tip:** You could display additional information in the partner list as well by selecting fields in the **Additional List Display Values** region.

34. Click **Submit**.

   The application returns you to the Fields page. Now that you’ve created object and its associated fields, you’ll next create the object’s work area, where your users can view and manage object records.

### Selecting an Icon

1. In the object tree, click the name of the Help Request object you’ve created.

   The Help Request: Overview page appears. A default icon appears for the object, but we’re going to change it.

2. Click **Change Icon**.

3. Select an icon from the available choices.

   The application returns you to the Overview page. Your chosen icon is now displayed.

### Creating the Default Pages

1. Click the **Pages** tree item.

   The application displays the Help Request: Pages page. You’ll begin by creating the default pages for your object.

2. Click the **Simplified Pages** tab if it is not already selected.

3. Click **Create Default Pages**.

   The application displays a set of default pages.

4. Click the **Default custom layout** link under **Search and Select Dialog Layouts**.

5. Click the **Edit** icon next to **Search Region**.

6. Select the fields to appear in the search region. Double-click each of the following list items to move them to the **Selected Fields** area:

   - Abstract
   - CreationDate
   - Status
   - Type

8. Click **Save and Close**.

9. Click the **Edit** icon next to **Search Results**.

10. In the Configure Search Results Table region, select the fields you want to appear in a local search. Double-click each of the following list items to move them to the **Selected Fields** area:

    - Abstract
    - CreationDate
    - Status
    - Type
11. Click **Save and Close**, then click **Done**.
12. Click the **Default custom layout** link under **Landing Page Layouts**.

   In the Configure Summary Table region, you will select the fields you want to appear in the summary table on the overview page of the application.

13. Click the **Edit** icon next to **Summary Table**.

   In the Configure Summary Table region, you will select the fields you want to appear in the summary table on the overview page of the application.

14. In the **Available Fields** region, double-click each of the following list items:
   - Company
   - Description
   - Creation Date
   - Created By
   - Status
   - Type

15. Click **Save and Close**, then click **Done**.
16. Click the **Default custom layout** link under **Creation Page Layouts**.
17. Click the **Edit** icon next to **Help Request Create**.
18. Double-click each of the following list items to select the fields you want to appear on the Create Help Request page:
   - Company
   - Description
   - Creation Date
   - Created By
   - Status
   - Type

19. Click **Save and Close**, then click **Done**.
20. Click the **Default custom layout** link under **Detail Page Layouts**.
21. Click the **Edit** icon next to **Summary**.
22. In the **Configure Detail Form** region, select the fields that will appear in the object’s default summary, which is the first region at the top of the details page. Double-click the list items so the following items appear in the Selected Fields region:
   - Abstract
   - Company
   - Status
   - Type
   - Description
   - Creation Date
   - Created By
   - Attachments
23. Click **Save and Close**, then click **Done**.

Now you’ll test your object’s pages.

**Testing the Object Pages**

1. Click the **Home** link, then click the **Help Requests** link.

   The application displays the help requests work area.

   Begin by creating a new help request.

2. Click the **Create** button.

   The Create Help Request page appears.

3. Enter “Laptop doesn’t work.” in the **Abstract** field.

4. Click the **Search: Company** link.

5. Select a company name and enter “Smoke is coming out of the back.” into the **Description** field.

6. Click the **Type** list.

7. Click the **Hardware** list item.

8. Click the **Save and Close** button.

   The application returns you to the work area Overview page. You will now review and update the help request you just created.

9. Click the **Laptop doesn’t work.** link.

   The Edit Help Request page appears showing the request you created. Now you’ll add an attachment.

10. Click the plus sign icon next to Attachments.

    The Attachments window appears.

11. Click the **Choose File** button and select any file to attach, then click **OK**.

12. Click the **Save and Close** button.
Adding Child or Related Objects: Example

This example demonstrates how you can continue to build out additional capabilities after building a new Help Request work area. You can use Application Composer to:

- Create a Solutions custom object, and set up a context link subtab for it in the Help Request work area. Users can create various solutions, and then link one or more solutions to a help request record.
- Create an Activity child object, and add a child subtab for it in the Help Request work area. Users can add activities that were executed for each help request.

Creating a Solutions Custom Object

1. Sign in to Oracle Sales Cloud and click the Navigator icon.

2. Click Customization under Tools, then select Application Composer.
3. Click the Application list.
4. Click the Sales list item.
5. In the Objects area, click the Create button.

6. Click in the Display Label field and enter “Solution”.
7. Click in the Plural Label field and enter “Solutions”.
The other fields are automatically populated with default values.

8. Click in the **Object Name** field and enter "Solution", then click OK.  
A new custom object named Solution is displayed.

9. Click the **Expand** button next to the Solution object.  
10. Click the **Fields** tree item.  
11. Click the **Create a custom field** button.

   **Note:** You might need to click the **Custom** tab first.

12. Click the **Long Text** option, then click **OK**.  
13. Click in the **Display Label** field and enter "Description".  
14. Click in the **Display Width** field and enter "100".  
15. Under Constraints, click the **Required** option.  
16. Click the **Updatable** option to select it if it is not already selected.  
17. Click the **Save and Close** button.  
18. Click the **Create a custom field** button.  
19. Click the **Choice List (Fixed)** option, then click **OK**.
20. Click in the **Display Label** field and enter "Type".
21. Under List of Values, click the **Search and Select Lookup Type** button.
22. Click in the **Lookup Type** field and enter "HR_TYPE".

23. Click the **Search** button, then click **OK**.
24. Under Appearance, click the **Multiple Select Choice List** option in the **Display Type** field.
25. Click the **Save and Close** button.
26. Under the Solution custom object, click the **Pages** tree item.
27. Click the **Create Work Area** link.
28. Click the **Menu Category** list.
29. Click the **Sales** list item.
30. Click in the **Menu Item Display Label** field and enter "Solutions".
31. Click the **Next** button.
32. In the **Configure Local Search** panel, double-click the **Solution Name** and **Type** list items to move them to Selected Fields.
33. Click the **Next** button.
34. In the **Configure Summary Table** panel, double-click the **Description** field and the **Type** list item.
35. In the **Configure Creation Page** panel, double-click the **Type** field.
36. Click the **Next** button.
37. In the **Configure Default Summary** panel, double-click the following fields and list items:
   - **Solution Name**
   - **Description**
   - **Type**
38. Add the **Attachments** field to the Selected Fields list in the Configure Detailed Summary section.
39. Click the **Save and Close** button.
40. Click the **Navigator** link.
41. Click the **Solutions** link.
42. Click the **Create** button.
43. Click in the **Solution Name** field and enter "Who uses a desktop?"
44. Click in the **Description** field and enter "Still many reasons for a desktop".
45. Click the **Type** list.
Click the Hardware list item, then click Save.

Click the Create button.

Click in the Solution Name field and enter "My laptop is on fire".

Click in the Description field and enter "Certain laptops are highly flammable".

Click the Type list.

Click the Hardware list item, then click Save.

Click the Create button.

Click in the Solution Name field and enter "Windows 7 tips and tricks".

Click in the Description field and enter "Download TweakUI".

Click the Type list.

Click the Software list item, then click Save.

Click the Create button.

Click in the Solution Name field and enter "Slow DSL Line".

Click in the Description field and enter "Review upgrade options."

Click the Type list.

Click the Hardware list item.

Click the Network list item, then click Save.

Next, you'll create a subtab in your help request object. Click the Navigator link.

Click the Customization link, then click the Application Composer link.

Click the Application list.

Click the Sales list item.

Click the Expand button next to Custom Objects, then click the Expand button next to the Help Request object.

Click the Pages tree item.

In the Subtabs region, click the Create button.

Click the Context link option, then click Next.

Click the Data Object list.

Click the Solution list item.

Click in the Display Label field and enter "Solutions".

Click the Field Name list.

Click the Type list item.

The Operator field is filled in automatically with the Equal to value.

Click the Value Type list, then click the Object field list item.
77. Click the **Source Object** list, then click the **Type** list item.

78. In the **Configure Summary Table** panel, double-click the following list items:
   - Description
   - Solution Name
   - Type

79. Click the **Save and Close** button.
80. Next, you'll test your subtab. Click the **Navigator** link.
81. Click the **Help Requests** link.
82. Click the **Laptop doesn't work** link.

   The Solutions tab is displayed, showing solutions that have the same Type as the help request.

83. Click the **Type** list.
84. Click the **Software** list item.
85. Click the **Save** button.

   The list of solutions changes to software solutions.
Creating an Activity Child Object

As Pinnacle users begin logging help requests, additional requirements surface: Can we start detailing the activities that we’re doing to resolve issues?

1. Click the Navigator link.
2. Click the Customization link, then click the Application Composer link.
3. Click the Application list.
4. Click the Sales list item.
5. Click the Help Request tree item.

Summary information about the object is displayed in the local area.

6. Click the Create Child Object button on the far right.

7. Click in the Display Label field and enter “Activity”.
8. Click in the Plural Label field.

The other fields on the page are automatically populated with default values. We’ll make changes to some of these values.

10. Click in the **Object Name** field and enter "Activity".

![Create Custom Child Object](image)

11. Click the **OK** button.

   The new Activity custom child object appears in the object tree.

![Activity: Overview](image)

12. Click the **Expand** button next to the **Activity** object.
13. Click the **Fields** tree item.
14. Click the **Create a custom field** button.
15. Click the **Long Text** option, then click **OK**.
16. Click in the **Display Label** field and enter "Description".
17. Click in the Display Width field and enter "100".
18. Click the Save and Close button.
19. Click the Create a custom field button.
20. Click the Choice List (Fixed) option, then click OK.
21. Click in the Display Label field and enter "Status".
22. Under List of Values, click the Search and Select Lookup Type button.
23. Click in the Lookup Type field and enter "HR_STATUS".

24. Click the Search button.
   The search result is displayed.
25. Click the OK button.
26. In the Default Value area, click the Fixed Value list.
27. Click the Open list item.
28. Click the Save and Close button.
   Now that we’ve created the Activity child object and its fields, let’s add the Activity subtab to the Help Request object.
29. Click the Pages button.
30. In the Subtabs region, click the Create button.

31. Child or related object should already be selected; if it is not, select it.
32. Click the Next button.
33. Click the Data Object list.
34. Click the Activity list item.
35. Click in the Display Label field and enter "Activities".
36. In the Configure Summary Table panel, double-click the following fields and list items:
   - Activity Name
   - CreationDate
   - Description
   - Status
37. In the Configure Detail Form panel, double-click the following fields and list items:
   - CreationDate
   - Description
   - Status
38. Click the Save and Close button.
39. Click the Navigator link.
40. Click the Help Requests link.
41. Click the Laptop doesn't work. link.
42. Click the Activities tab.
43. Click the Add Row button.
44. Click in the Activity Name field and enter "Reviewed options with customer."
45. Click in the Description field and enter "Recommended fire extinguisher."
46. Click the Add Row button.
47. Click in the Activity Name field and enter "Call in experts?"
48. Click in the Description field and enter "Possibly the fire department could help."
49. Click the Save and Close button.

You have successfully added child and related objects.
Advanced Field and Object Properties: Example

As the value of the Help Request system becomes clear, management asks the Oracle Sales Cloud administrator to enforce business policies:

- Don’t allow activities to be added after a help request is closed.
- Provide more granularity for how help requests are categorized.
- Make sure people follow up on activities within a week.
- Offer summary information about activity volume for each help request.

This example has four sections:

- Add a trigger to the Activity object.
- Add a dependent choice list to the Help Request object.
- Add a due date with field validation.
- Add a Total Activities aggregate function.

Adding a Trigger to the Activity Object

Management asks the administrator to enforce business policies. The first request is to not allow activities to be added after a help request is closed. In this section, we’ll add a trigger to the activity object so that a new record cannot be created when the parent help request status = ‘Closed’.

1. Click the Application list, then click the Sales list item.
2. Expand Custom Objects, then expand Activity.
3. Click the Server Scripts link.
4. Click the Triggers tab.
5. Click the Add a new Trigger object.

6. Enter "OnlyIfOpenHR" in the Trigger Name field.
7. Enter "Prevent creation of an activity if the help request is closed." in the Description field.
8. Click the **Expand Show Palette** button.

9. Click the **Fields** tab.
10. Click the **Expand** link next to Activity on the **Fields** tab.
11. Click the Help Request object.

12. Click the **Status** object.

   ✏️ **Note:** You might need to scroll to see it.

13. Enter `if` in the Expression field.
14. With **Status** selected, click the Insert button.

   The application inserts code that identifies the **Status** field.

15. Enter `== 'Closed'`{  
16. Enter `adf.error.raise(null)` in the Expression field.
17. Enter `} in the Expression field.

   Your completed expression should look like this:
   
   ```
   if (HelpRequest_c?.Status_c=='Closed')
   { adf.error.raise(null) }
   ```

18. Click the **Validate** button.

   A warning appears; you can safely ignore it in this case.

19. Click the **Expand Error Message** link.
20. Enter "You cannot create a new activity on a closed help request." in the Error Message field.
21. Click the **Save and Close** link.
22. Click the **Navigator** link.
23. Click the **Help Requests** link.
24. Click the **Laptop doesn't work.** link.
25. Click the Status list, then click the **Closed** list item.
26. Click the **Save** button.
27. In the Activity subtab, click the Add Row button.

   The application displays your error message because you attempted to add a new activity to a help request with a status of Closed.

28. Click the OK button.

29. Click the Status list, then click the Open list item.

30. Click the Save and Close button.

Adding a Dependent Choice List

Now sales management wants to provide better granularity for how help requests are categorized. In this section, we’ll create an Area fixed choice list field on the Help Request object and constrain its possible values by the selected Type field value.

1. Navigate to Application Composer.

2. Click the Application list, then click the Sales list item.

3. Click the Expand button next to the Help Request object.

4. Click the Fields tree item.

5. Click the Create a custom field button.

6. Click the Choice List (Fixed) option, then click OK.

7. Enter "Area" in the Display Label field.

8. Press [Tab].

   The application populates the Name field with a default value, which we will accept.

9. In the List of Values area, click the Create a New Lookup Type button.

10. Enter “XY Help Request Area” in the Meaning field.

11. Enter “XY_HR_AREA” in the Lookup Type field.

12. Click the Create Lookup Code button, and enter the following information. Click the Create Lookup Code button after entering the information for each line in the table:

<table>
<thead>
<tr>
<th>Value in Meaning Field</th>
<th>Value in Lookup Code Field</th>
<th>Value in Display Sequence Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-ROM</td>
<td>CD-ROM</td>
<td>10</td>
</tr>
<tr>
<td>Desktop</td>
<td>Desktop</td>
<td>20</td>
</tr>
<tr>
<td>Laptop</td>
<td>Laptop</td>
<td>30</td>
</tr>
<tr>
<td>Router</td>
<td>Router</td>
<td>40</td>
</tr>
<tr>
<td>Application</td>
<td>Application</td>
<td>50</td>
</tr>
<tr>
<td>Operating System</td>
<td>Operating System</td>
<td>60</td>
</tr>
</tbody>
</table>
13. Click the **Save** button.

14. In the List of Values area, click the **Constrain list by parent field value selection** option.

15. Click the Parent Choice List.

16. Click the **Type** list item.

17. Click the **Create a New Value Map** button.

   First, we'll select lookups for the Hardware type, which is selected by default.

18. Double-click the following list items:
   - Desktop
   - Laptop
   - CD-ROM
19. Click the **Software** object.
20. Double-click the **Application** and **Operating System** list items.

21. Click the **Network** object.
22. Double-click the **Application** and **Router** list items.

23. Click the **OK** button.
24. Click the **Save and Close** button.
25. Click the **Pages** tree item.
26. Click the **Edit Creation Page** link.
27. Double-click the **Area** list item to move it to the **Selected Fields** list.
28. Click the **Save and Close** button.
29. Click the **Edit Summary Form** link.
30. Double-click the **Area** list item to move it to the **Selected Fields** list.
31. Click the **Save and Close** button.
32. Click the **Navigator** link.
33. Click the **Help Requests** link.
34. Click the **Laptop doesn't work**. link.
35. Click the **Type** list.
36. Click the **Hardware** list item.
37. Click the **Area** list.
Notice the items that appear in the Area list.

38. Click the Laptop list item.
39. Click the Save button.
40. Click the Type list.
41. Click the Software list item.
42. Click the Area list.

Notice that the items displayed are different from those you see when you selected Hardware in the Type field.

43. Click the Application list item.
44. Click the Type list.
45. Click the Network list item.
46. Click the Area list.
Notice that the Area list now shows network-specific items.

47. Click the Router list item.
48. Click the Save and Close button.

Adding a Due Date with Field Validation

The help request system has been in place long enough that the focus is moving from logging and tracking to speed of resolution. Our latest requirement is to ensure that people follow up on activities within a week. Our Oracle Sales Cloud administrator implements this enhancement quickly and easily. In this section, we'll create a due date/time field on the Activity object, and create a field validation rule to ensure that an activity’s due date is within the next seven days.

1. Navigate to Application Composer.
2. Click the Application list, then click the Sales list item.
3. Expand the Custom Objects item, then expand the Activity object.
4. Click the Fields tree item.
5. Click the Create a custom field button.
6. Click the Datetime option.
7. Click the OK button.
8. Enter "Due" in the Display Label field.

Notice that the application automatically provides a default value for the Name field.

9. In the Default Value area, click the Expression option.
10. Click the Enter the expression used for the field's default value button.
11. Click the \textbf{Date} object, then click the \textbf{today} list item under Function.

12. Click the \textbf{Insert} button.
13. Enter "+1" into the \textbf{Expression} field, then click \textbf{OK}.

14. Click the \textbf{Save and Close} button.
15. Under the Activity object, click the \textbf{Server Scripts} tree item.
16. Click the \textbf{Validation Rules} tab.
17. Under Field Rules, click the \textbf{Add a new validation rule} button.
18. Click the \textbf{Field Name} list.
19. Click the **Due** list item.

20. Enter “futureDate” in the **Rule Name** field.

21. Enter “Due date must be in the next week.” in the **Description** field.

22. Under Rule Definition, click the **Expand Show Palette** button.

23. Click the **Keywords** tab.

24. Click the **newValue** object.

25. Click the **Insert** button.

26. Click the **>=** list item.

27. Click the **Functions** tab.

28. In the Category column, click the **Date** object.
29. In the Function column, click the `today` object.

![Oracle Sales Cloud Functions Palette]

30. Click the Insert button.
31. Click the `&` button.
32. Enter `newValue <= today() + 7.` into the Expression field.

33. Click the Validate button.

A message is displayed, indicating that the expression parsed successfully.

34. Enter "Due date must be in the future but no more than one week from today." in the Error Message field.
35. Click the Save and Close button.
36. Expand the Help Request object.
37. Click the Pages tree item.
38. In the Subtabs panel, click the Activities object.
39. Click the Edit button.
40. In the Configure Summary Table area, double-click the Due list item to move it to the Selected Fields list.
41. In the Configure Detail Form area, double-click in the Due field to move it into the Selected Fields list.
42. Click the Save and Close button.
43. Click the Navigator link.
44. Click the Help Requests link.
45. Click the Laptop doesn't work link.
46. The Activities tab should be selected by default; if it is not, select it.
47. Click the Add Row button.
48. Enter "Reviewed options with customer." in the Activity Name field.
49. Enter "Recommended fire extinguisher." in the Description field.
50. At the far right of the **Description** field, click the **Select Date and Time** icon.

![Image of Description field and Due date field with date 2/14/14 12:00 AM]

51. For the purposes of this activity, we will assume today is May 18. Click May 28 on the calendar.

52. Click the **OK** button, then click **Save**.

The error message that you created earlier appears, because May 28 is more than a week after May 18.

![Image of error message with date 5/28/14 12:00 AM]

53. Now we’ll return the Due date to a valid value. To the right of the **Due** field, click the **Select Date and Time** button.

54. Click May 23.

55. Click the **OK** button, then click **Save** twice.

The date can be saved now, because it is within the allowed range.

56. Click the **Save and Close** button.

### Adding a Total Activities Aggregate Function

Finally, we’ll provide summary information about activity volume for each help request. In this last section, we’ll create a formula field that leverages an aggregate function to count the number of activities under a help request.

1. Navigate to Application Composer.
2. Click the **Application** list, then click the **Sales** list item.
3. Under the Help Request object, click the **Fields** tree item.

   **Note:** If necessary, expand the Help Request object first.

4. Click the **Create a custom field** button.
5. Click the **Formula** option, then click **OK**.
6. Click the **Number** option.
7. Enter “Total Activities” in the **Display Label** field.
8. Enter “20” in the **Display Width** field.
9. Click the **Next** button.
10. Click the **Expand Show Palette** button.
11. In the Category column, click the **Number** list item.

12. In the Function column, click the **count** list item.

13. Click the **Insert** button.
14. Click the **Activities** tree item.

15. Click the **CreatedBy** field.

   ✍️ **Note:** You can click any field in this list.

16. Click the **OK** button, then click **Submit**.
17. In the Objects list, click the vertical scrollbar.
18. Under Help Request, click the **Pages** tree item.
19. The Desktop Pages tab should be selected; if it isn't, click it to select it.
20. Click the **Edit Summary Form** link.
21. In the Configure Default Summary panel, double-click the **Total Activities** list item to move it to the **Selected Fields** list.
22. Click the **Save and Close** button.
23. Click the **Navigator** link.
24. Click the Help Requests link.
25. Click the Laptop doesn't work link.

You should see the Total Activities field, showing the total number of activities associated with the help request (in our case it is 3, as you can see by looking at the list of activities under the Activities tab).

Adding Subtabs: Example

After building the new Help Request work area, you can continue to build out additional capabilities:

- Add a common components subtab to track notes
- Add a Web content subtab based on a URL, to show relevant information about our partners, including their Web sites

In this example, you will:

- Use Application Composer to add a Notes subtab to the Help Requests work area
- Add a Web content subtab

Adding a Notes Subtab

1. Sign in to Oracle Sales Cloud and navigate to Application Composer.
2. Click the Application list, then click the Sales list item.
3. Expand the Custom Objects object, then expand the Help Request object.
4. Click the Pages tree item.
5. Under Subtabs, click the Create button.

6. Click the Common component option, then click Next.
7. On the Edit Subtab: Common Component page, click the **Notes** option.

**Edit Subtab: Common Component**

Configure Common Component Child Objects

Add common components as subtabs to an object's details page.

- Notes [x]
- Tasks [ ]
- Interactions [ ]
- Appointments [ ]

8. Click the **Save and Close** button.
9. Click the **Navigator** link.
10. Click the **Help Requests** link.
11. Click the **Laptop doesn't work.** link.
12. Click the **Notes** subtab.
13. On the Notes subtab, click the **Create** button.
14. Enter "This is an unusual situation." in the **Create Note** field.

15. Click the **OK** button, then click **Save and Close**.

Adding a Web Content Subtab

You will now add a URL field to the Help Request object, which will be used to display Web pages on a Web content subtab. End users can use this to view relevant information about partners, including Web sites.

1. Navigate to Application Composer.
2. Click the **Application** list, then click the **Sales** list item.
3. Expand the Custom Objects object, then expand the Help Request object.
4. Click the **Fields** tree item.
5. Click the **Create a custom field** button.
6. Click the **Formula** option, then click **OK**.
   The Text option is selected by default.
7. Enter "Company URL" in the **Display Label** field.
8. Click the **Depends On** list.
9. Click the **Company** list item, then click **Next**.
10. In the Text Value area, click the **Expand Show Palette** button.
11. Click the **Fields** tab.
12. Expand the **Help Request** object.
13. Click the **Company** tree item.

14. Click the **Sort Descending** button.

**Note:** You might need to hover your mouse over the **Display Label** header to see the sort buttons.

15. Scroll and click the **Primary Web Site** object.
16. Click the **Insert** button.

17. Click the **Submit** button.
18. Click the **Pages** tree item.

19. Under Creation Page, click the **Edit Creation Page** link.

20. Double-click the **Company URL** list item to move it to the Selected Fields area.

21. Click the **Save and Close** button.

22. Under Subtabs, click the **Create** button to create a new subtab.

23. Click the **Web content** option.

24. Click the **Next** button.

25. Enter "Partner Web Site" in the **Display Label** field.

26. Under URL Definition, click the **Expand Show Palette** button.

27. Click the **Fields** tab.
28. Click the **Company URL** object.

29. Click the **Insert** button, then click **Save and Close**.
30. Click the **Navigator** link.
31. Click the **Help Requests** link.
32. Click the **Laptop doesn't work.** link.
33. Click the **Partner Web Site** tab.

The Web site is displayed on the Partner Web Site subtab.

**Testing Custom Object Security Settings: Example**

By default, a custom object and its records are visible and editable only to a default duty role specified by the application. Application Composer provides a set of security policy configuration pages that lets you grant both functional and data access to custom objects.

In this example, you will grant additional access to your two custom objects, Help Request and Activity, using Application Composer’s security policy configuration pages. This example demonstrates how to do the following:

- Test the visibility of the Help Request work area that you previously created.
- Change the security settings of the Help Request object, so that others can view the object’s associated work area.
- Check the visibility of our Help Request work area, now that we’ve modified the security settings on the Help Request object for the sales manager.
• Change the security settings of the Activity child object so that others can view the Activities subtab on the Help Request work area.
• Check the visibility of our Activities subtab in the Help Request work area, now that we’ve modified the security settings on the Activity child object for the sales manager.

Testing the Visibility of the Help Request Work Area

1. Sign in to Oracle Sales Cloud as a sales representative or sales manager user.
2. Click the Navigator link.

   Notice that the Help Requests link does not appear, because your user doesn’t have permission to access this information.
3. Click the Sign Out link.

Changing the Help Request Object Security Settings

1. Sign back in as an Oracle Sales Cloud application administrator. Click the Sign In button.
2. Click the Navigator link.
3. Click the Customization link under Tools, then click the Application Composer link.
4. Click the Application list, then click the Sales list item.
5. Expand the Custom Objects item, then scroll and expand the Help Request object.
6. Click the Security link.
7. Scroll and select the following options for the Sales Manager duty option by selecting their check boxes:
   - Create
   - View
   - Update
   - Delete
   - View All
   - Update All

   Notice that the Create and Delete items have been deselected.
8. Click the **Create** and **Delete** items to re-select them.

9. Click the **Save and Close** button.
10. Click the **Sign Out** link.

Checking the Visibility of the Help Request Work Area

1. Sign back in as a sales manager.
2. Click the **Navigator** link.
3. Click the **Help Requests** link.
4. Click the **Laptop doesn't work** link.

   Notice that nothing appears under the Activities tab (which is displayed by default) because your user does not have permission to view this information.
5. Click the **Sign Out** link.

Changing the Activity Object Security Settings

1. Sign back in as an Oracle Sales Cloud application administrator. Click the **Sign In** button.
2. Click the **Navigator** link.
3. Click the **Customization** link under Tools, then click the **Application Composer link**.
4. Click the **Application** list.
5. Click the **Sales list** item.
6. Click the **Role Security** tree item.

7. Click the **Sales Manager Duty** object.

8. Click the **Define Policies** button.

9. Select the Access check boxes for the following options in the Activity_c row, clicking the Activity_c option after each:
   - View
   - Update
   - Create
   - Delete

10. Click the **View All** and **Update All** check boxes in the Activity_c row.

    Notice that the **Create** and **Delete** check boxes have been deselected.
11. Click the **Create** and **Delete** check boxes to re-select them.

12. Click the **Save and Close** button.

13. Click the **Sign Out** link.

Checking the Visibility of the Activities Subtab

1. Sign back in as a sales manager.
2. Click the **Sign In** button.
3. Click the **Navigator** link.
4. Click the **Help Requests** link.
5. Click the **Laptop doesn't work** link.

Notice that the Activities tab now shows content, and you can create new Activities.
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.

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.

customization
A change to the predefined artifacts of the application. Customizations impact multiple users.

customization layer
A level that represents the scope of users that a customization impacts. 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.

**design time**
The type of activities that developers perform at the code or data model level.

**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.

**flexfield**
A flexible data field that you can customize to contain one or more segments or store 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 area**
The region at the very top of 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 area 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.

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
Customizations made in this layer affect all users.

springboard
The grid of icons on the home page or the strip of icons above all simplified pages. Use the icons to open pages.

task flow infolet
An infolet that displays summary information about a task.

user layer
Customizations 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.